Theoretical Computer Science 431 (2012) 96–116

Contents lists available at SciVerse ScienceDirect

Theoretical Computer Science

journal homepage: www.elsevier.com/locate/tcs

Foundational aspects of multiscale modeling of biological systems with process algebras

Roberto Barbuti [a], Giulio Caravagna [a], Andrea Maggiolo-Schettini [a], Paolo Milazzo [a,*], Simone Tini [b]

[a] Dipartimento di Informatica, Università di Pisa, Largo Pontecorvo 3, 56127 Pisa, Italy

[b] Dipartimento di Scienze e Alta Tecnologia, Università dell'Insubria, Via Carloni 78, 22100 Como, Italy

ARTICLE INFO

Keywords:
Process algebra
Multiscale modeling
Structural operational semantics
Bisimulations

ABSTRACT

We propose a variant of the CCS process algebra with new features aiming at allowing multiscale modeling of biological systems. In the usual semantics of process algebras for modeling biological systems actions are instantaneous. When different scale levels of biological systems are considered in a single model, one should take into account that actions at a level may take much more time than actions at a lower level. Moreover, it might happen that while a component is involved in one long lasting high level action, it is involved also in several faster lower level actions. Hence, we propose a process algebra with operations and with a semantics aimed at dealing with these aspects of multiscale modeling. We give both a reduction semantics and an SOS semantics for our new algebra with a result of operational correspondence between the two. Moreover, we study behavioral equivalences for such an algebra and give some examples.

1. Introduction

Formal modeling notations of computer science are nowadays often applied to the description of biological systems. Such notations can be used to unambiguously describe the structure and the events governing the dynamics of the systems of interest, thus allowing development of analysis tools such as simulators and of formal analysis techniques based, for instance, on model checking or on behavioral equivalences.

As examples of formalisms that have been applied to the description of biological systems, we mention Bio-PEPA [22,23], the (stochastic) π-calculus [44,45,47], Bioambients [46], Stochastic Bioambients [18], the κ-calculus [25], the Language for Biological Systems (LBS) [42], the Continuous π-calculus [33] and the Calculus of Looping Sequences (CLS) [37,7,9]. In these formalisms the dynamics of a biological system consists of a sequence of events (usually biochemical reactions) described either as communications between processes of a process algebra, or as applications of some rewrite rules. In the stochastic extension of these formalisms, the dynamics of a system is described by taking also into account the different rates of occurrence of the events. Rates depend on some parameters associated with the events (e.g. kinetic constants of the corresponding biochemical reactions) and on the abundance (or concentration) of the entities (or reactants) that can cause such events. The rates are then used, as in Gillespie's algorithm [30], to describe both the exponentially distributed time elapsing between two subsequent events, and the probability of an occurring event.

This way of describing the dynamics of biological systems with sequences of events, however, assumes that the occurrence of one of such events can be described as an instantaneous change in the system state. In fact, even in the

* Corresponding author. Tel.: +39 0502213144.
E-mail addresses: barbuti@di.unipi.it (R. Barbuti), caravagn@di.unipi.it (G. Caravagna), maggiolo@di.unipi.it (A. Maggiolo-Schettini), milazzo@di.unipi.it (P. Milazzo), simone.tini@uninsubria.it (S. Tini).

0304-3975/$ – see front matter © 2011 Elsevier B.V. All rights reserved.
doi:10.1016/j.tcs.2011.12.058

stochastic approach, the only notion of time that is considered is given by the frequency of the events, rather than by their duration. The duration of an event is usually ignored if it is negligible with respect to the time interval between two events, or hidden in such a time interval by choosing a rate for the event that is small enough to take into account both its frequency and its duration.

Phenomena of interest in the study of biological systems often include processes at different levels of abstraction. A typical example is cell signaling, that involves gene regulation and protein interaction processes at the intra-cellular level, a signal diffusion process at the inter-cellular level, and some macroscopic change at the tissue level. The processes at the different levels influence each other. However, they involve system components of very different sizes and are characterized by events having very different durations. This motivates the *multiscale approach to modeling*, whose application to biological systems seems to be promising [4,2,14,19,48,49].

The description of the dynamics of biological systems by means of sequences of instantaneous events is not suitable for multiscale models. When different scale levels of biological systems are considered in a single model, one should take into account that events at a level may take much more time than events at a lower level. Moreover, it might happen that while a component (e.g. a cell) is involved in a long lasting high level event (e.g. mitosis), it is involved also in several faster lower level events (e.g. protein synthesis) that neither consume such a component nor interrupt the higher level event. Consequently, the distinction between the time scales of the events, the possibility of having the same component involved in several events at different time scales, and the fact that the completion of some events may (or may not) interfere with other events in which the same component is involved require new notions of system dynamics to be considered.

In this paper, we propose a process algebra with operations and with a semantics aimed at dealing with these aspects of multiscale modeling. We aim at undertaking a foundational study. Hence, we consider a minimal process algebra for the description of biological systems (a fragment of CCS [38] and of the Chemical Ground Form [21]) and we make the minimal changes we think to be necessary to describe the new aspects of interest.

As regards the syntax of the new process algebra, called *Process Algebra with Preemptive and Conservative actions (PAPC)*, we propose a new action prefixing operator that allows an action to be executed in a *conservative* (or non-consuming) manner, namely without removing the process that performed it. As regards the semantics, we define it as a labeled transition system by following the *ST semantics* approach ([35,31,34,17]) in which actions are not instantaneous, but described by two separate starting and ending transitions. This will allow for processes in which multiple actions are running in parallel and competing for their completion. Indeed, we change the usual interpretation of the summation operator (by making it slightly similar to a parallel composition) in order to allow a process to be involved in several actions at the same time. The termination of an action in a summation may interrupt (in a *preemptive* way) the others that are concurrently executed in the same summation, depending on which action prefixing operator is used.

We shall define both a reduction semantics for PAPC and an SOS semantics. The former semantics is rather simple and describes the possible behaviors of a process as a whole. The latter semantics is a compositional semantics that allows us to study behavioral equivalences. A theorem of operational correspondence between the two semantics is proved. Moreover, we define a notion of bisimulation for the process algebra and we prove a congruence result for it. Some examples are given of use of the process algebra and of the bisimulation relation.

The paper is structured as follows. In Section 2 we introduce the syntax of PAPC, the reduction semantics and an example of modeling. In Section 3 we define the compositional semantics, prove the result on operational correspondence and study a bisimulation relation. Finally, we end with some conclusions and discussion of further work in Section 4.

This paper is an extended and revised version of [8]. With respect to [8] in the present paper we have slightly changed the definition of the syntax of PAPC, we have added the reduction semantics (with the results on the operational correspondence with the compositional semantics) and a new example of modeling. Moreover, some of the concepts developed in this paper have been studied in [20].

2. A process algebra with preemptive and conservative actions

In this section we present the syntax of a *Process Algebra with Preemptive and Conservative actions*, denoted as PAPC. We present also a reduction semantics for such an algebra and an example of modeling.

2.1. Syntax

PAPC is a process algebra with dyadic communication in the style of CCS [38] and of the π-calculus [39,40]. Hence, communications involve exactly two processes at a time. As in the cases of CCS and of the π-calculus, the description of biological systems with PAPC is based on a *processes-as-molecules* view [21,47]: the modeling of a molecular species occurring n times in a system includes n copies of a process modeling a single molecule.

Differently with respect to the approach of classical process algebras, where actions are instantaneous, in PAPC we assume that actions can consume time and that the instants of start and completion of an action can be detached. An action that has already started but not yet completed is said to be *running*.

Let us assume an infinite set of process constants $\mathbb{C}$ ranged over by $A, B, C, \ldots$, an infinite set of actions Act ranged over by $\alpha, \beta, \ldots$ and a total function $\bar{\ } : Act \to Act$ such that $\overline{\overline{\alpha}} = \alpha$. We also denote with Act_τ the set of actions enriched with the special internal action $\tau \notin Act$, $Act_\tau = Act \cup \{\tau\}$. The abstract syntax of PAPC is a follows.

Definition 1. *(Guarded) summations* S and *Processes* P of PAPC are given by the following grammar:

$$S ::= \mathbf{0} \mid \alpha.P \mid \alpha : (S, P) \mid S + S \mid A$$
$$P ::= S \mid P \mid P$$

where $\alpha \in Act$ and $A \in \mathbb{C}$. We denote the set of all summations as $\mathcal{S}$ and the set of all processes as $\mathcal{P}$.

As usual, $\mathbf{0}$ denotes the classical idle process that can perform no action. PAPC processes can perform actions in Act in two different ways, represented by two different action prefixing operators. Both $\alpha.P$ and $\alpha : (S, P)$ can perform the action α. However, after performing such an action, the first process simply behaves as P, while the second process continues its execution as S and produces a copy of P. This allows $\alpha.P$ to model a molecule that can be involved in a reaction that transforms it into another molecule, and $\alpha : (S, P)$ to model a molecule that can be involved in a reaction that does not consume it. Another difference between $\alpha.P$ and $\alpha : (S, P)$ is related with the fact that actions are not instantaneous and that a process (more precisely a summation) can start several actions concurrently. When an action used as in $\alpha.P$ completes, it interrupts all other running actions of the summation containing it. This agrees with the intuition that $\alpha.P$ represents a transformation of the described molecule into a different one. On the other hand, when an action used as in $\alpha : (S, P)$ completes, it does not interrupt the other running actions of the summation containing it. Again, this agrees with the intuition that the described molecule is neither consumed nor transformed into something different. We say that α is *preemptive* when used as $\alpha.P$ and that it is *conservative* when used as $\alpha : (S, P)$. Notice that in PAPC the composition of these two approaches is possible, since an action α may be conservative for a process, and the complementary action $\overline{\alpha}$ may be preemptive.

A process $S_1 + S_2$ is able to start actions of both S_1 or S_2, but notice that, because of the difference between conservative and preemptive actions and the fact that actions are not instantaneous, this summation operator does not correspond to the choice operator of classical process algebras. In fact, starting an action in S_1 (resp. S_2) does not imply that S_2 (resp. S_1) is discarded. As a consequence, $S_1 + S_2$ may start several actions, which are said to be in *competition*, meaning that they run concurrently until one of them completes. When an action α in S_1 (resp. S_2) completes, then all actions in S_2 (resp. S_1) competing with it have to be interrupted only if α is preemptive. If this is the case, then S_2 (resp. S_1) is discarded. Differently, if a conservative action completes, then the summation is not discarded and, consequently, all the running actions in the summation are still running.

A process is either a summation S or a parallel composition of processes $P_1 \mid P_2$. Handshaking is possible between any action α in P_1 and its complementary action $\overline{\alpha}$ in P_2. When handshaking is performed, an unique identifier is assigned to the instance of communication and may be used to interrupt the communication. An interesting case here is when an action α in P_1 and its complementary action $\overline{\alpha}$ in P_2 have been coupled with an handshaking, have started and not yet terminated, and some preemptive action in P_1 competing with α completes, so that α must be interrupted. In fact, in this case also the complementary action $\overline{\alpha}$ in P_2 has to be interrupted.

Finally, process constants (such as A) are used to specify recursive systems. In general, systems are specified as a set of constant defining equations of the form $A \stackrel{\text{def}}{=} S$. Note that we restrict the use of constants at the level of summations (rather than at the level of processes) since a process is always a parallel composition of a finite number of summations, hence a definition $A \stackrel{\text{def}}{=} P$ can always be replaced by n definitions $A_1 \stackrel{\text{def}}{=} S_1, \ldots A_n \stackrel{\text{def}}{=} S_n$ such that $P = S_1 \mid \ldots \mid S_n$.

Some further considerations are worth in order to introduce the differences between PAPC and the classical process algebras. The capability of having competing actions is at the basis for the choices we made in the definition of PAPC. A summation is a process that can start *multiple actions in parallel*, but can be involved in *each action at most once at a time*. Notice that, in classical process algebras, this is not possible since an action, when starts, determines the future process to transform the summation in. Hence, the choice is resolved at the time of the starting of an action. In this sense, the summation operator of PAPC is not a classical choice for the reason that the competing actions compete for their completion, then the semantics of the completion, and hence the semantics of the PAPC summation, will depend on the type of the action to be completed, namely whether it is conservative or preemptive.

Consequently, at any time of a computation a summation could be in a configuration in which some of its actions are currently running. More precisely, the competition of the running actions is due to the fact that they are waiting to complete. Practically, the time for completion may be modeled by general distributions as in [17], or by delays as in [5,6,20]. However, in this definition of the algebra we do not consider quantitative timing and stochasticity.

Summing up, the features that makes PAPC a process algebra able to deal with some aspects of multiscale modeling are the distinction between preemptive and conservative actions, the capability to observe starting and ending of actions and a new summation operator. A different approach to multiscale modeling is that of Process Algebra with Hooks (PAH) [26,27], where processes are explicitly allocated at several *scales*, i.e. abstraction levels, and two different operators are offered for process synchronization: one to synchronize processes at the same level, the other to synchronize processes at different levels. Actually, to model a system component with different behavioral levels, PAPC requires one process in which the activities at different levels are combined with PAPC summation operator, whereas PAH requires one process for each level. On one side, PAH permits us to have a cleaner image about the different scale levels, on the other side PAPC permits higher

level actions to interrupt lower level ones without requiring synchronization with them, which seems to be more natural. Overall, the two approaches may be combined.

It is worth noting that PAPC supports multiscale modeling, but not impose this style of modeling. In particular, it does not require fast actions to be conservative (as it often happens in multiscale models). Indeed, it is equally possible to construct models in which fast actions are preemptive. This indicates that PAPC could be used for types of modeling beyond biological.

Finally, some comments are needed about how PAPC deals with interrupts. Following [24], process algebras with notions of priority between actions deal with both *must preemption*, when the *availability* of an action with higher priority preempts actions with lower priority, and *may preemption*, when the *execution* of an action with higher priority preempts actions with lower priority. In PAPC we have may preemption, but note that the event that causes it is not the execution of an action but its termination.

In order to define the semantics of PAPC we need to model a process with possibly running actions. We do this by introducing a notion of process configuration.

Definition 2. *Summation configurations SC* and *process configurations C* of PAPC are given by the following grammar:

$$SC ::= \mathbf{0} \mid \alpha.P \mid \alpha : (S,P) \mid [\alpha]^l.P \mid [\alpha]^l : (S,P) \mid SC + SC \mid A$$
$$C ::= SC \mid C \mid C$$

where $\alpha \in Act$, $A \in \mathbb{C}$, $l \in \mathbb{N}$, $S \in \mathcal{S}$ and $P \in \mathcal{P}$. We denote the set of all summation configurations as $\mathcal{SC}$ and the set of all process configurations as $\mathcal{C}$.

Note that the set of all processes corresponds to the set of all process configurations that contain no $[\alpha]^l$ prefixes, hence $\mathcal{P} \subset \mathcal{C}$. Consequently, in what follows we will define relations on process configurations and use them also on processes. The same relationship holds for summations and summation configurations, namely $\mathcal{S} \subset \mathcal{SC}$.

With respect to a process, a process configuration may contain actions denoted by a different prefix. In particular, the configuration $[\alpha]^l.P$ is the configuration reached by $\alpha.P$ after α has started, and $[\alpha]^l : (S,P)$ is the configuration reached by $\alpha : (S,P)$ after α has started. For both the action prefixes, the new argument $l \in \mathbb{N}$ is a natural number that identifies a pair of running actions which started together. Notice that these identifiers, which have to be unique, are computed by the handshaking performed before the start of an action and, once a preemptive action is completed, they may be used to interrupt all other competing actions. Then, if one of these competing actions is α and it has started an handshaking with another action $\overline{\alpha}$ in a process running in parallel, then also this action $\overline{\alpha}$ will be interrupted. This can be obtained by assigning the same identifier to these two actions when the handshaking begins.

2.2. Reduction semantics

We define a reduction semantics for PAPC, namely a semantics given essentially as a transition relation between PAPC process configurations, and which is closed with respect to a structural congruence that equates configurations that are syntactically different, but should be interpreted as the same configuration.

We start with the definition of the structural congruence, which is rather standard.

Definition 3. The *structural congruence* is the least congruence $\equiv$ on PAPC process configurations which satisfies the following axioms:

$$SC_1 + SC_2 \equiv SC_2 + SC_1 \qquad SC + \mathbf{0} \equiv SC$$
$$(SC_1 + SC_2) + SC_3 \equiv SC_1 + (SC_2 + SC_3) \qquad A \equiv S \ \text{ if } A \stackrel{\text{def}}{=} S$$
$$C_1 \mid C_2 \equiv C_2 \mid C_1 \qquad C \mid \mathbf{0} \equiv C \qquad (C_1 \mid C_2) \mid C_3 \equiv C_1 \mid (C_2 \mid C_3).$$

Structural congruence states (i) commutativity and associativity of both $+$ and $|$, (ii) that $\mathbf{0}$ is the neutral element of both $+$ and $|$, and (iii) that a process constant A can be replaced with the summation S used in its definition.

Now, we define by structural recursion an auxiliary function $Id : \mathcal{C} \mapsto \wp(\mathbb{N})$ as follows:

$$Id([\alpha]^l.P) = Id([\alpha]^l : (S,P)) = \{l\}$$
$$Id(SC_1 + SC_2) = Id(SC_1) \cup Id(SC_2)$$
$$Id(C_1 \mid C_2) = Id(C_1) \cup Id(C_2)$$
$$Id(\mathbf{0}) = Id(\alpha.P) = Id(\alpha : (S,P)) = Id(A) = \emptyset.$$

The value $Id(C)$ denotes the set of the identifiers of the actions in the configuration C that are running. For instance, given a configuration $C = [\alpha]^l.P_1 + \beta : (S,P_2) \mid [\gamma]^{l'}.P_3$, the identifiers collected by function Id are given by $Id(C) = \{l, l'\}$.

We exploit structural congruence and the Id function to define *well-formedness* and *completeness* of PAPC process configurations as follows.

Definition 4. A PAPC process configuration $C \in \mathcal{C}$ is *well-formed and complete* if and only if for every P_1, SC_1, S_1 and C' such that either

$$C \equiv [\alpha]^l.P_1 + SC_1 \mid C' \quad \text{or} \quad C \equiv [\alpha]^l : (S_1, P_1) + SC_1 \mid C'$$

it holds $l \notin Id(SC_1)$ and either

$$C' \equiv [\overline{\alpha}]^l.P_2 + SC_2 \mid C'' \quad \text{or} \quad C' \equiv [\overline{\alpha}]^l : (S_2, P_2) + SC_2 \mid C''$$

for some $C'' \in \mathcal{C}$ with $l \notin Id(C'')$ and $l \notin Id(SC_2)$.

Moreover, $C \in \mathcal{C}$ is *well-formed* (but not necessarily complete) if and only if there exists $C' \in \mathcal{C}$ such that $C \mid C'$ is well-formed and complete.

A process configuration C is well-formed if for each running action α with identifier l it contains, either there is no other running action in C with the same identifier, or there is exactly one running action $\overline{\alpha}$ in C with the same identifier. If the latter case holds for all running actions in C, then C is also complete. Obviously, any process $P \in \mathcal{P}$ is a well-formed and complete process configuration.

The following lemma states a property of well-formed and complete process configurations that will be used in proofs of theorems.

Lemma 1. *Given a well-formed and complete PAPC process configuration $C \in \mathcal{C}$ such that either $C \equiv [\alpha]^l.P_1 + SC_1 \mid C'$ or $C \equiv [\alpha]^l : (S_1, P_1) + SC_1 \mid C'$ with P_1, SC_1, S_1 and C' as in Definition 4, the following equalities hold:*

1. $Id(SC_1) \cap Id(SC_2) = Id(SC_1) \setminus Id(C'')$
2. $Id(SC_1) \cap Id(C'') = Id(SC_1) \setminus Id(SC_2)$.

Proof. Identifiers in a well-formed and complete process configuration must occur exactly twice. Moreover, SC_1 cannot contain two instances of the same identifier. The second occurrences of the identifiers appearing in SC_1 can be either in SC_2 or in C''. Consequently, $Id(SC_1) \subseteq Id(SC_2) \cup Id(C'')$. The two equalities of the lemma follow from this and from the fact that, by definition of well-formed and complete process configuration, it holds $Id(SC_1) \cap Id(SC_2) \cap Id(C'') = \emptyset$. □

We define a function $Reset(C, L)$, where $L \subseteq \mathbb{N}$, which gives a process configuration C' such that all running actions in C associated with identifiers in L are substituted with the corresponding non-running actions. This represents a form of rollback activity that makes the involved actions available to start again. Formally, the function $Reset(C, L)$ is recursively defined as follows:

$$Reset([\alpha]^l.P, L) = \begin{cases} \alpha.P & \text{if } l \in L \\ [\alpha]^l.P & \text{if } l \notin L \end{cases}$$

$$Reset([\alpha]^l : (S, P), L) = \begin{cases} \alpha : (S, P) & \text{if } l \in L \\ [\alpha]^l : (S, P) & \text{if } l \notin L \end{cases}$$

$$Reset(SC_1 + SC_2, L) = Reset(SC_1, L) + Reset(SC_2, L)$$
$$Reset(C_1 \mid C_2, L) = Reset(C_1, L) \mid Reset(C_2, L)$$
$$Reset(\alpha.P, L) = \alpha.P$$
$$Reset(\alpha : (S, P), L) = \alpha : (S, P)$$
$$Reset(A, L) = A$$
$$Reset(\mathbf{0}, L) = \mathbf{0}.$$

We give also a lemma on a property of the function *Reset* that will be useful in the following.

Lemma 2. *Given any PAPC process configuration C and any set of identifiers $L \subseteq \mathbb{N}$, it holds: $Reset(C, L) = Reset(C, L \cap Id(C))$.*

Proof. Trivial. □

The reduction semantics is defined as a labeled transition system on process configurations. The semantics is in the ST style, hence labels are of the forms $l+$ and $l-$ with $l \in \mathbb{N}$. We denote with L^+ and L^-, respectively, the sets of all labels in the two forms, namely $L^+ = \{l+ \mid l \in \mathbb{N}\}$ and $L^- = \{l- \mid l \in \mathbb{N}\}$. Let ℓ range over $L^+ \cup L^-$.

Definition 5. The *reduction semantics* of PAPC is the labeled transition system $(\mathcal{C}, L^+ \cup L^-, \xrightarrow{\ell})$ where $\xrightarrow{\ell} \subseteq \mathcal{C} \times (L^+ \cup L^-) \times \mathcal{C}$ is the least labeled transition relation on process configurations closed with respect to $\equiv$ and satisfying the following inference

rules:

$$(+1)\ \frac{l = \min(\mathbb{N} \setminus (Id(SC_1) \cup Id(SC_2) \cup Id(C)))}{\alpha.P_1 + SC_1 \mid \overline{\alpha}.P_2 + SC_2 \mid C \xrightarrow{l+} [\alpha]^l.P_1 + SC_1 \mid [\overline{\alpha}]^l.P_2 + SC_2 \mid C}$$

$$(+2)\ \frac{l = \min(\mathbb{N} \setminus (Id(SC_1) \cup Id(SC_2) \cup Id(C)))}{\alpha.P_1 + SC_1 \mid \overline{\alpha} : (S, P_2) + SC_2 \mid C \xrightarrow{l+} [\alpha]^l.P_1 + SC_1 \mid [\overline{\alpha}]^l : (S, P_2) + SC_2 \mid C}$$

$$(+3)\ \frac{l = \min(\mathbb{N} \setminus (Id(SC_1) \cup Id(SC_2) \cup Id(C)))}{\alpha : (S_1, P_1) + SC_1 \mid \overline{\alpha} : (S_2, P_2) + SC_2 \mid C \xrightarrow{l+} [\alpha]^l : (S_1, P_1) + SC_1 \mid [\overline{\alpha}]^l : (S_2, P_2) + SC_2 \mid C}$$

$$(-1)\ \frac{L = Id(SC_1) \cup Id(SC_2) \quad C' = Reset(C, L)}{[\alpha]^l.P_1 + SC_1 \mid [\overline{\alpha}]^l.P_2 + SC_2 \mid C \xrightarrow{l-} P_1 \mid P_2 \mid C'}$$

$$(-2)\ \frac{L = Id(SC_1) \quad SC_2' = Reset(SC_2, L) \quad C' = Reset(C, L)}{[\alpha]^l.P_1 + SC_1 \mid [\overline{\alpha}]^l : (S, P_2) + SC_2 \mid C \xrightarrow{l-} P_1 \mid S + SC_2' \mid P_2 \mid C'}$$

$$(-3)\ \frac{}{[\alpha]^l : (S_1, P_1) + SC_1 \mid [\overline{\alpha}]^l : (S_2, P_2) + SC_2 \mid C \xrightarrow{l-} S_1 + SC_1 \mid S_2 + SC_2 \mid P_1 \mid P_2 \mid C}.$$

Rules $(+1)$, $(+2)$ and $(+3)$ describe a handshaking between actions α and $\overline{\alpha}$ in a PAPC process configuration. In particular, rule $(+1)$ describes the case in which both actions are used as preemptive actions, rule $(+2)$ describes the case in which one is used as a preemptive action and the other as a conservative action, and rule $(+3)$ describes the case in which both are used as conservative actions. All of the three rules simply transform α and $\overline{\alpha}$ into $[\alpha]^l$ and $[\overline{\alpha}]^l$, respectively, where l is the least identifier that is not used in the whole process configuration.

Rules (-1), (-2) and (-3) describe a completion of actions that previously performed a handshaking. Again, rule (-1) describes the case in which both actions are used as preemptive actions, rule (-2) describes the case in which one is used as a preemptive action and the other as a conservative action, and rule (-3) describes the case in which both are used as conservative actions. In the case of (-1) we have that all actions in C that have previously performed a handshake with some component in SC_1 or SC_2 have to be unlocked, and this is what function *Reset* does. Note that since both α and $\overline{\alpha}$ are used as preemptive actions, both SC_1 and SC_2 disappear in the configuration reached by the described transition (as in the usual semantics of choice in CCS-like process calculi). In the case of (-2) action $\overline{\alpha}$ is used as a conservative action, hence S replaces $\overline{\alpha} : (S, P_2)$ in the summation and SC_2 is not removed. Moreover, in this case *Reset* is used to rollback both actions in SC_2 and actions in C which previously performed a handshaking with some actions in SC_1. Furthermore, process P_2 is composed in parallel with the rest of the configuration reached by the transition. Finally, in the case of (-3) no action has to be unlocked, hence S_1, S_2, P_1 and P_2 are simply placed in the proper places inside the configuration reached by the transition without the need of using *Reset*.

2.3. An example

In order to show how PAPC can be used to model biological systems we apply it to the description of an example of tissue development [32,36].

A tissue is an ensemble of cells that together carry out a specific function (giving a shape to an organ, contracting a muscle, transmitting signals from nervous system, etc…). Some tissues are continuously growing (such as in plants) whereas others take a permanent shape. In any case cells of a tissue can die or being damaged, and a loss of cells in a tissue should be compensated with some tissue repairing process. Tissue repairing consists of creation of new cells through divisions of the available tissue cells. This process is usually regulated by means of some inter-cellular communication process. For instance, a cell of a tissue might be able to sense the presence of other cells by means of some cell adhesion molecules (proteins located on the cell surface and involved with the binding with other cells) that may inhibit cell proliferation. Additionally, a cell may stimulate proliferation of other cells in the tissue by diffusing some growth factor (proteins whose presence in the environment can be sensed by cells and be interpreted as a stimulus to grow).

When a cell interprets signals from the other cells of the tissue as a stimulus to grow, it starts a duplication process known as "cell cycle". The cell cycle consists of four phases: G_1, S, G_2 and M. Phases G_1 and G_2 are gap (or resting) phases. In phase S (synthesis) the main event which happens is the replication of DNA. In the last phase M (mitosis) the cell segregates the duplicated sets of chromosomes between daughter cells and then divides into two cells. The duration of the cell cycle depends on the type of cell (e.g a human normal cell takes approximately 24 hours to perform a cycle).

Now we give a PAPC process modeling an abstract tissue. The model will contain events at two different scale levels: we have events at the *intra-cellular* level, namely the passage from one phase of the cell cycle to another, and events at the *inter-cellular* (or *tissue*) level, namely cell duplications and communications between cells.

We start with some process definitions:

$$Cell \stackrel{\text{def}}{=} Stop + \overline{dying}.\mathbf{0} + dying.(Cell_{G1} \mid Clock) \qquad Stop \stackrel{\text{def}}{=} \overline{stop} : (Stop, \mathbf{0})$$
$$Cell_{G1} \stackrel{\text{def}}{=} G_1 + stop.Cell + dying : (D, \mathbf{0}) \qquad G_1 \stackrel{\text{def}}{=} g_1 : (Cell_S, \mathbf{0})$$
$$Cell_S \stackrel{\text{def}}{=} S + stop.Cell + dying : (D, \mathbf{0}) \qquad S \stackrel{\text{def}}{=} s : (Cell_{G_2}, \mathbf{0})$$
$$Cell_{G2} \stackrel{\text{def}}{=} G_2 + stop.Cell + dying : (D, \mathbf{0}) \qquad G_2 \stackrel{\text{def}}{=} g_2 : (Cell_M, \mathbf{0})$$
$$Cell_M \stackrel{\text{def}}{=} M + stop.Cell + dying : (D, \mathbf{0}) \qquad M \stackrel{\text{def}}{=} m : (Cell_{G_1}, Cell_{G1} \mid Clock \mid Clock)$$
$$Clock \stackrel{\text{def}}{=} \overline{g_1}.\bar{s}.\overline{g_2}.\overline{m}.\mathbf{0} \qquad D \stackrel{\text{def}}{=} dying : (D, \mathbf{0}).$$

Cell describes a tissue cell in a situation that is perceived as stable, namely there is no need of new cells. In this situation the cell communicates to other cells (usually in the neighborhood) that they can stop replicating, and this is modeled by the $\overline{stop}$ action. Moreover, a cell can die, and this is modeled by $\overline{dying}$ action that is used to allow other cells to sense its absence. (For the sake of simplicity we assume that the signal is sent to only one other cell.) Consequently, a cell can also perform action *dying* to perceive the death of a cell in the neighborhood. These three actions are composed as a summation in which $\overline{dying}$ and *dying* are used as preemptive actions whereas $\overline{stop}$, which does not change the state of the cell, is used as a conservative action.

If the cell performs action *dying*, it starts the cell cycle (in phase G_1). A cell in a phase X of the cell cycle is described by process $Cell_X$, and a cell can pass through the four phases G_1, S, G_2 and M by performing actions g_1, s, g_2 and m, respectively (see definitions of processes G_1, S, G_2 and M). These four actions can be performed by synchronizing with the auxiliary process *Clock*, a copy of which is created every time a cell enters the cell cycle (note that after mitosis two copies of *Clock* are necessary for the two resulting cells).

A cell in any phase of the cell cycle can receive a stop signal (by performing the *stop* action) and this causes the cell cycle to be interrupted since *stop* is used as a preemptive action with *Cell* as continuation. Dying signals received by cells involved in the cell cycle are ignored.

According to the reduction semantics a single instance of *Cell* cannot perform any transition. However, if we have n cells one of the possible behavior of the system is described by the following sequence of transitions (where $n \times Cell$ stands for $Cell \mid \ldots \mid Cell$ with n instances of *Cell*):

$$\begin{aligned}
n \times Cell &\xrightarrow{1+} (n-2) \times Cell \mid \overline{stop} : (Stop, \mathbf{0}) + \left[\overline{dying}\right]^1.\mathbf{0} + dying.(Cell_{G1} \mid Clock) \\
&\quad \mid \overline{stop} : (Stop, \mathbf{0}) + \overline{dying}.\mathbf{0} + [dying]^1.(Cell_{G1} \mid Clock) \\
&\xrightarrow{1-} (n-2) \times Cell \mid Cell_{G1} \mid Clock \\
&\xrightarrow{1+} (n-2) \times Cell \mid [g_1]^1 : (Cell_S, \mathbf{0}) + stop.Cell + dying : (D, \mathbf{0}) \mid [\overline{g_1}]^1.\bar{s}.\overline{g_2}.\overline{m}.\mathbf{0} \\
&\xrightarrow{2+} (n-3) \times Cell \mid \overline{stop} : (Stop, \mathbf{0}) + \left[\overline{dying}\right]^2.\mathbf{0} + dying.(Cell_{G1} \mid Clock) \\
&\quad \mid [g_1]^1 : (Cell_S, \mathbf{0}) + stop.Cell + [dying]^2 : (D, \mathbf{0}) \mid [\overline{g_1}]^1.\bar{s}.\overline{g_2}.\overline{m}.\mathbf{0} \\
&\xrightarrow{2-} (n-3) \times Cell \mid [g_1]^1 : (Cell_S, \mathbf{0}) + stop.Cell + dying : (D, \mathbf{0}) \mid [\overline{g_1}]^1.\bar{s}.\overline{g_2}.\overline{m}.\mathbf{0} \\
&\xrightarrow{1-} (n-3) \times Cell \mid Cell_S \mid \bar{s}.\overline{g_2}.\overline{m}.\mathbf{0} \\
&\xrightarrow{1+}\xrightarrow{1-} (n-3) \times Cell \mid Cell_{G2} \mid \overline{g_2}.\overline{m}.\mathbf{0} \\
&\xrightarrow{1+}\xrightarrow{1-} (n-3) \times Cell \mid Cell_M \mid \overline{m}.\mathbf{0} \\
&\xrightarrow{1+}\xrightarrow{1-} (n-3) \times Cell \mid Cell_{G1} \mid Cell_{G1} \mid Clock \mid Clock \\
&\xrightarrow{1+} (n-4) \times Cell \mid \left[\overline{stop}\right]^1 : (Stop, \mathbf{0}) + \overline{dying}.\mathbf{0} + dying.(Cell_{G1} \mid Clock) \\
&\quad \mid G_1 + [stop]^1.Cell + dying : (D, \mathbf{0}) \mid Cell_{G1} \mid Clock \mid Clock \\
&\xrightarrow{1-} (n-2) \times Cell \mid Cell_{G1} \mid Clock \mid Clock \\
&\xrightarrow{1+}\xrightarrow{1-} (n-1) \times Cell \mid Clock \mid Clock.
\end{aligned}$$

This sequence of transitions describes the behavior in which a cell dies, and this stimulates another cell to enter the cell cycle. While the latter cell is in phase G_1 of the cell cycle, another cell dies by sending a signal to the very same cell (already in the cell cycle). The signal is hence ignored and the cell cycle continues until phase M is reached that causes two new cells in phase G_1 to be obtained. Finally, some of the $(n-3)$ cells that have not yet performed any action sends a stop signals to

the two cells in phase G_1 causing the cell cycle to be interrupted for both of them. In the end we have $(n-1)$ cells and two instances of *Clock* that are a garbage left by cells whose cell cycle has been interrupted.

In this abstract model of tissue development cells can non-deterministically die or communicate with each other. A precise description of the behavior should be obtained by adding quantitative aspects to the model, namely by specifying rates of events and probability distributions for the durations of the performed actions, and by adding some spatial information to capture which cell can communicate with other cells.

2.4. Properties of the reduction semantics

It is easy to see that well-formedness and completeness of process configurations are preserved by the reduction semantics of PAPC.

Proposition 1. *Given a well-formed and complete process configuration $C \in \mathcal{C}$, if either $C \xrightarrow{l+} C'$ or $C \xrightarrow{l-} C'$, then C' is well-formed and complete as well.*

Proof. Follows trivially from the fact that rules of the reduction semantics either add a pair of coupled actions with a fresh identifier (rules $(+1)$, $(+2)$ and $(+3)$), or remove one of such pairs (rules (-1), (-2) and (-3)). □

The same property holds for well-formedness alone, but it is less interesting since, as a consequence of Proposition 1, non-complete configurations cannot be reached if the initial state of the system is, as expected, a process $P \in \mathcal{P}$.

We conclude the treatment of the reduction semantics for PAPC by showing that if we assume actions to be instantaneous, namely that each $\xrightarrow{l+}$ transition is immediately followed by a $\xrightarrow{l-}$ transition, we obtain a semantics that is analogous to the standard one for CSS.

First of all, let us define a CSS-like reduction semantics for PAPC. Note that in this case the semantics is a transition system in which states are processes, rather than configurations.

Definition 6. The *instantaneous reduction semantics* of PAPC is the transition system $(\mathcal{P}, \rightarrow)$ where $\rightarrow \subseteq \mathcal{P} \times \mathcal{P}$ is the least transition relation on processes closed with respect to $\equiv$ and satisfying the following inference rules:

$$(inst\,1)\ \frac{}{\alpha.P_1 + S_1 \mid \overline{\alpha}.P_2 + S_2 \mid P \rightarrow P_1 \mid P_2 \mid P}$$

$$(inst\,2)\ \frac{}{\alpha.P_1 + S_1 \mid \overline{\alpha} : (S_2', P_2) + S_2 \mid P \rightarrow P_1 \mid S_2' + S_2 \mid P_2 \mid P}$$

$$(inst\,3)\ \frac{}{\alpha : (S_1', P_1) + S_1 \mid \overline{\alpha} : (S_2', P_2) + S_2 \mid P \rightarrow S_1' + S_1 \mid P_1 \mid S_2' + S_2 \mid P_2 \mid P}.$$

We can now give the following result on the correspondence between the instantaneous reduction semantics and the previously given reduction semantics.

Proposition 2. *Given two PAPC processes P and P', it holds*

$$P \rightarrow P' \text{ if and only if } P \xrightarrow{1+} C \xrightarrow{1-} P' \text{ for some } C \in \mathbb{C}.$$

Proof. Both implications $P \rightarrow P'$ implies $P \xrightarrow{1+} C \xrightarrow{1-} P'$ and $P \xrightarrow{1+} C \xrightarrow{1-} P'$ implies $P \rightarrow P'$ can be trivially proved by cases on the inference rules of the two semantics and by relating rule $(instX)$ with $(+X)$ and $(-X)$, with $X \in \{1, 2, 3\}$. □

3. Structural operational semantics and bisimulations for PAPC

In this section we define a Structural Operational Semantics (SOS) [43] and study bisimulation relations for PAPC.

3.1. Structural operational semantics

The aim of the SOS is to equip PAPC with a Labeled Transition System (LTS), namely a set of transitions of the form $C \xrightarrow{\ell}_r C'$ representing a move from $C \in \mathcal{C}$ to $C' \in \mathcal{C}$, with the *label* ℓ carrying some information about the move and the index r used to group transitions describing a particular aspect of the behavior of the processes. The LTS is defined by a set of SOS transition rules of the form $\frac{\text{premises}}{\text{conclusion}}$. Intuitively, each of these rules explains how a move of a process is obtained from moves of its subprocesses. All our rules are in Figs. 1–5. We assume the standard way for assigning an LTS with such a set of transition rules (see, e.g., [1]). Labels of transitions are structured in several components. This is needed to build transitions compositionally, as argued, e.g., in [10–13].

The main features of the SOS we want are the following. Firstly, it must have a mechanism to interrupt competing actions and this mechanism is activated by the completion of a preemptive action. Secondly, the style of the semantics must be ST-like, as this permits to easily observe detached events as the start and the completion of an action.

$$(H_1)\ \ \alpha.P \xrightarrow{1,\alpha^+}_H [\alpha]^1.P \qquad (H_2)\ \ \alpha : (S,P) \xrightarrow{1,\alpha^+}_H [\alpha]^1 : (S,P)$$

$$(H_3)\ \ \frac{CS_1 \xrightarrow{l,\alpha^+}_H CS_1' \qquad l \notin Id(CS_2) \qquad \alpha \in Act}{CS_1 + CS_2 \xrightarrow{l,\alpha^+}_H CS_1' + CS_2}$$

$$(H_4)\ \ \frac{CS_1 \xrightarrow{l,\alpha^+}_H CS_1' \qquad l \in Id(CS_2) \quad l' = \min(\mathbb{N} \setminus Id(CS_1 + CS_2)) \qquad \alpha \in Act}{CS_1 + CS_2 \xrightarrow{l',\alpha^+}_H CS_1'[l'/l] + CS_2}$$

$$(H_5)\ \ \frac{C_1 \xrightarrow{l,\alpha^+}_H C_1' \qquad l \notin Id(C_2) \quad \alpha \in Act_\tau}{C_1 \mid C_2 \xrightarrow{l,\alpha^+}_H C_1' \mid C_2}$$

$$(H_6)\ \ \frac{C_1 \xrightarrow{l,\alpha^+}_H C_1' \qquad l \in Id(C_2) \quad l' = \min(\mathbb{N} \setminus Id(C_1 \mid C_2)) \quad \alpha \in Act_\tau}{C_1 \mid C_2 \xrightarrow{l',\alpha^+}_H C_1'[l'/l] \mid C_2}$$

$$(H_7)\ \ \frac{C_1 \xrightarrow{l,\alpha^+}_H C_1' \qquad C_2 \xrightarrow{l',\overline{\alpha}^+}_H C_2' \quad l'' = \min(\mathbb{N} \setminus Id(C_1 \mid C_2)) \qquad \alpha \in Act}{C_1 \mid C_2 \xrightarrow{l'',\tau^+}_H C_1'[l''/l] \mid C_2'[l''/l']}$$

$$(H_8)\ \ \frac{S \xrightarrow{\alpha,l^+}_H SC \qquad \alpha \in Act}{A \xrightarrow{\alpha,l^+}_H SC} \qquad \text{if } A \stackrel{\text{def}}{=} S$$

Fig. 1. The handshaking relation $\rightarrow_H \subseteq \mathcal{C} \times \Theta^+ \times \mathcal{C}$.

In order to get these features, we define a relation for modeling the start of an action and the coupling of processes; this will be named as the *handshaking relation*. Furthermore, we define a *completion relation* for modeling the finishing of both preemptive and conservative actions. These two relations will make use of an *interruption relation* to model the interruption of currently running actions, as required by the notion of preemptive actions.

The handshaking relation

This relation is used to model the starting of an action and the coupling of the processes starting complementary actions. The handshaking relation is $\rightarrow_H \subseteq \mathcal{C} \times \Theta^+ \times \mathcal{C}$, where Θ^+ contains labels θ^+ of the form

$$\theta^+ = (l, \alpha^+)$$

where $l \in \mathbb{N}$ represents the identifier assigned to the started action $\alpha \in Act_\tau$, and the use of the superscript "+" comes from the definition of the semantics in the ST style, in order to denote the start of an action. The SOS rules in Fig. 1 are at the basis of the definition of $\rightarrow_H$. We implicitly assume the rules symmetric to (H_3), (H_4), (H_5), (H_6).

Rules (H_1) and (H_2) model the starting of an action α. At any time a process with prefix α can start action α moving to a configuration in which it cannot perform the same action anymore, i.e. the configuration $[\alpha]^1.P$ or, analogously, the configuration $[\alpha]^1 : (S,P)$. Such a configuration, together with the one describing the process performing the complementary action, has to be uniquely identified by a natural number representing the identifier of the just started action. At this step, the process simply chooses 1 as unique identifier. All our choices for assigning identifiers to actions are inspired by those of [16], which ensure that the obtained LTS is finite-branching. The rules for binary operators + and | will solve conflicts of colliding identifiers, if any. Notice that both preemptive actions and conservative actions start in the same way.

Rules (H_3) and (H_4) combine the start of an action with operator +. In rule (H_3) the identifier l of the action α started by summation configuration SC_1 has no conflicts with the identifiers of the competing actions running in summation configuration SC_2. Differently, in the case of rule (H_4) a conflict does exist, which implies that a fresh identifier l' replaces l. Again, the policy by which we choose the new fresh identifier, along the line of [16], is such that the resulting LTS is finite-branching.

Rules (H_5), (H_6) and (H_7) combine the start of an action with the operator |. Rules (H_5) and (H_6) model an autonomous move by one of the two processes C_1 and C_2, and deal with identifiers as (H_3) and (H_4), respectively. Rule (H_7) models a handshaking performed by two processes able to perform complementary actions. As in classical process algebras, we do not force C_1 and C_2 to handshake, since C_1 could handshake with a further process composed in parallel with $C_1 \mid C_2$.

Notice that here we may have a conflict even if in C_2 the action associated with the colliding identifier is the complementary action $\overline{\alpha}$. Rule (H_7) models the handshaking by assigning to this particular instance of synchronization a new fresh identifier l'' chosen with the same policy used to resolve conflicts in the previous rules. The renaming of both old

$$(I_1)\ \ \mathbf{0} \xrightarrow{\emptyset}_I \mathbf{0} \qquad (I_2)\ \ A \xrightarrow{\emptyset}_I A$$

$$(I_3)\ \ [\alpha]^l.P \xrightarrow{\{l\}}_I \alpha.P \qquad (I_4)\ \ [\alpha]^l : (S,P) \xrightarrow{\{l\}}_I \alpha : (S,P)$$

$$(I_5)\ \ [\alpha]^l.P \xrightarrow{\emptyset}_I [\alpha]^l.P \qquad (I_6)\ \ [\alpha]^l : (S,P) \xrightarrow{\emptyset}_I [\alpha]^l : (S,P)$$

$$(I_7)\ \ \alpha.P \xrightarrow{\emptyset}_I \alpha.P \qquad (I_8)\ \ \alpha : (S,P) \xrightarrow{\emptyset}_I \alpha : (S,P)$$

$$(I_9)\ \ \frac{SC_1 \xrightarrow{L}_I SC_1' \qquad SC_2 \xrightarrow{M}_I SC_2'}{SC_1 + SC_2 \xrightarrow{L\cup M}_I SC_1' + SC_2'} \qquad (I_{10})\ \ \frac{C_1 \xrightarrow{L}_I C_1' \qquad C_2 \xrightarrow{M}_I C_2'}{C_1 \mid C_2 \xrightarrow{L\cup M}_I C_1' \mid C_2'}$$

Fig. 2. The interruption relation $\rightarrow_I \subseteq \mathcal{C} \times \wp(\mathbb{N}) \times \mathcal{C}$.

identifiers with the newly generated is due to the fact that, in general, the two processes will have two different candidate identifiers, i.e. l and l'. The system in this case exhibits the internal action τ^+. By applying this rule, the two processes terminated this *handshaking phase*.

Finally, rule (H_8) is the standard rule for recursive definitions.

The interruption relation

This relation, which is not standard in classical process algebras, captures the identifiers of the actions that are running in a configuration and that must be interrupted. In the meantime, these actions are rolled-back so that they can start again. The interruption is caused by the completion of some competing preemptive actions. The interruption relation is $\rightarrow_I \subseteq \mathcal{C} \times \wp(\mathbb{N}) \times \mathcal{C}$, where a label $M \in \wp(\mathbb{N})$ contains the identifiers of the actions that have been interrupted. The rules presented in Fig. 2 are at the basis of the definition of $\rightarrow_I$.

Rules (I_1) and (I_2) simply state that $\mathbf{0}$ and a constant A cannot interrupt any action.

At any time, a process either in configuration $[\alpha]^l.P$ or $[\alpha]^l : (S,P)$ may interrupt the action it is currently performing. In these cases, treated with rules (I_3) and (I_4), it moves to a configuration in which the interrupted action α may start again, namely to configuration $\alpha.P$ or $\alpha : (S,P)$, respectively. In both the rules, the identifier l of the interrupted action is exhibited as the label of this transition. Again, this information will be used to interrupt also the partner of this action, as we are assuming that there is a partner in the system which, after the handshaking phase, has been coupled with the same label l.

In some cases not all of the actions have to be interrupted, so the processes in configuration $[\alpha]^l.P$ or $[\alpha]^l : (S,P)$ must be able also to non-deterministically decide whether to interrupt or not. This second case is described by rules (I_5) and (I_6), which may seem controversial at first glance. In particular, it may not be clear why a process may independently decide whether to interrupt or not some of the currently running actions. The need of this autonomy for the process can be clarified by an example. Let us assume a process configuration $(SC_1 + SC_2) \mid (SC_3 + SC_4) \mid C_1 \mid C_2$, where both SC_1 and SC_3 successfully complete a preemptive action. The actions to be interrupted are those currently running in both SC_2 and SC_4, namely those with identifiers denoted by $Id(SC_2) \cup Id(SC_4)$. Let us assume that some of the actions that have to be interrupted in SC_2 and SC_4 were coupled with some actions in C_1. In this case, also these actions in C_1 should be interrupted as well. Moreover, C_1 may be involved in other actions currently running and coupled with actions in C_2. Indeed, these actions must not be interrupted. This means that from C_1 the correct derivation with the interruption relation, in general, will not exhibit as label $Id(C_1)$, indeed it will exhibit a strict subset of $Id(C_1)$. This implies that C_1 must be able to autonomously decide which actions to interrupt, and this can be done by properly combining derivations of the interruption relation. The composition of the relations of the whole semantics will provide the correctness, namely the fact that all and only those to interrupt are actually interrupted.

Also, a process which is not performing any action, namely a process in a configuration $\alpha.P$ or $\alpha : (S,P)$, does not interrupt any action, as stated by rules (I_7) and (I_8).

Finally, rules (I_9) and (I_{10}) simply collect the labels of the interrupted actions in a summation and in a parallel composition, respectively.

The completion relation for preemptive actions

This relation is used to model the completion of a preemptive action. We will define completion relations also for conservative actions as well as the combination of both preemptive and conservative actions.

The completion relation for preemptive actions is $\rightarrow_{CP} \subseteq \mathcal{C} \times \Theta^-_{CP} \times \mathcal{C}$, with Θ^-_{CP} containing labels of the form

$$\theta^- ::= (l, \alpha^-, L_1, L_2)$$

where $l \in \mathbb{N}$ represents the identifier that was assigned to the completed action $\alpha \in Act_\tau$ when it was started, $L_1 \in \wp(\mathbb{N})$ and $L_2 \in \wp(\mathbb{N})$ are the sets of the identifiers of the actions that are interrupted by the termination of α and $\overline{\alpha}$, respectively, and the

$$(CR_1)\quad [\alpha]^l.P \xrightarrow{l,\alpha^-,\emptyset,\emptyset}_{CP} P \qquad (CR_2)\quad \frac{SC_1 \xrightarrow{l,\alpha^-,L_1,\emptyset}_{CP} P_1'}{SC_1 + SC_2 \xrightarrow{l,\alpha^-,L_1\cup Id(SC_2),\emptyset}_{CP} P_1'}$$

$$(CR_3)\quad \frac{C_1 \xrightarrow{l,\alpha^-,L_1,L_2}_{CP} C_1' \qquad C_2 \xrightarrow{M}_{I} C_2' \qquad (L_1 \cap Id(C_2)) \subseteq M \qquad L_2 \cap Id(C_2) = \emptyset \qquad l \notin Id(C_2) \qquad \alpha \in Act_\tau}{C_1 \mid C_2 \xrightarrow{l,\alpha^-,L_1\setminus M,L_2\cup(M\setminus L_1)}_{CP} C_1' \mid C_2'}$$

$$(CR_4)\quad \frac{C_1 \xrightarrow{l,\alpha^-,L_1,L_2}_{CP} C_1' \qquad C_2 \xrightarrow{l,\overline{\alpha}^-,M_1,M_2}_{CP} C_2' \qquad L_2 \subseteq M_1 \qquad M_2 \subseteq L_1 \qquad (M_1 \cap Id(C_1)) \subseteq (L_1 \cup L_2) \qquad (L_1 \cap Id(C_2)) \subseteq (M_1 \cup M_2)}{C_1 \mid C_2 \xrightarrow{l,\tau^-,(L_1\setminus(M_1\cup M_2))\cup(M_1\setminus(L_1\cup L_2)),\emptyset}_{CP} C_1' \mid C_2'}$$

Fig. 3. The completion relation for preemptive actions $\rightarrow_{CP} \subseteq \mathcal{C} \times \Theta^-_{CP} \times \mathcal{C}$.

use of the superscript "−" comes from the definition of the semantics in the ST style. More precisely, the set L_1 contains the identifiers of all running actions that are in the summation containing the completed action α. On the other hand, L_2 contains identifiers of actions contained in process configurations that are composed in parallel with the summation containing the completed action α. Actions with identifiers in L_2 are assumed to be involved in synchronizations with running actions in the summation containing $\overline{\alpha}$. For example, if $C = [\overline{a}]^1.P_1 + [b]^2 : (S_2, P_2) \mid [c]^3.P_3$, then any transition $C \xrightarrow{1,\overline{a},L_1,L_2}_{CP} C'$ has $L_1 = \{2\}$, since b is interrupted by the completion of $\overline{a}$, and $L_2 \subseteq \{3\}$, since c may be interrupted by the completion of a if the corresponding $\overline{c}$ action is in the same summation as a.

The rules presented in Fig. 3 are at the basis of the definition of $\rightarrow_{CP}$. We implicitly assume rules symmetric to (CR_2) and (CR_3).

Rule (CR_1) describes the completion of a preemptive action. When it completes, as the action is preemptive, the process is substituted by its continuation P. In the label, the identifier l is needed to couple this process with the one performing the corresponding complementary action $\overline{\alpha}$, which will have the same identifier l because of the handshaking, and the two $\emptyset$ state that no action is interrupted.

Rule (CR_2) states that the completion of a preemptive action in SC_1 affects a summation $SC_1 + SC_2$ so that all actions running in SC_2 should be interrupted. This is obtained by adding the set of actions currently running in the process SC_2 to the set L_1 of identifiers of actions interrupted by α. Hence the exhibited set of labels becomes $L \cup Id(SC_2)$. Since this rules deals with summations only, and not with parallel compositions, the last element of the transition label is left empty.

Rule (CR_3) states that the completion of a preemptive action in C_1 affects a parallel composition $C_1 \mid C_2$ so that all actions running in C_2 that are coupled with actions interrupted in C_1, must be interrupted as well. This rule describes the case in which the $\overline{\alpha}$ action coupled with the considered α action is not in C_2, namely $l \notin Id(C_2)$. This implies that $L_2 \cap Id(C_2) = \emptyset$, namely the identifiers of actions assumed in C_1 to be interrupted by $\overline{\alpha}$ are not in C_2. The actions that are interrupted by C_1 are contained in L_1. Actions running in C_2 that are coupled with actions interrupted in C_1 are hence $L_1 \cap Id(C_2)$. In order to ensure that all of them are interrupted it is required that $(L_1 \cap Id(C_2)) \subseteq M$. In the conclusion of the rule we have that the set of identifiers of actions interrupted by α becomes $L_1 \setminus M$, namely actions interrupted by α whose coupled actions are in C_2 are removed from this set. Moreover, the set of identifiers of actions assumed to be interrupted by $\overline{\alpha}$ becomes $L_2 \cup (M \setminus L_1)$, namely it now includes also the actions that are reset in C_2 by assuming that the coupled actions are interrupted by $\overline{\alpha}$.

Rule (CR_4) models the case in which both C_1 and C_2 complete preemptive actions that were coupled. As in classical process algebras, the whole system $C_1 \mid C_2$ exhibits an internal action τ. In the conclusion of the rule we have that the set of identifiers of actions interrupted corresponds to the union of L_1 and M_1, but in which coupled interrupted actions between C_1 and C_2 are removed. The set of identifiers of actions assumed to be interrupted in the context of C_1 and C_2 becomes $\emptyset$.

The completion relation for conservative actions

This relation is used to model the completion of a conservative action. This relation is $\rightarrow_{CC} \subseteq \mathcal{C} \times \Theta^-_{CC} \times \mathcal{C}$, where Θ^-_{CC} contains labels of the form

$$\theta^- ::= (l, \alpha^-, L, P)$$

where $l \in \mathbb{N}$ represents the identifier assigned to the completed action $\alpha \in Act_\tau$, $L \in \wp(\mathbb{N})$ is the set of identifiers of the interrupted actions, and $P \in \mathcal{P}$ is the new process created by the action which terminated and that, syntactically, must be propagated at the level of a parallel composition. At first sight it could sound strange that we need the component L. The idea

$$(CR_5)\quad [\alpha]^l : (S,P) \xrightarrow{l,\alpha^-,\emptyset,P}_{CC} S \qquad (CR_6)\quad \frac{SC_1 \xrightarrow{l,\alpha^-,L,P}_{CC} SC_1' \quad SC_2 \xrightarrow{M}_{I} SC_2'}{SC_1 + SC_2 \xrightarrow{l,\alpha^-,L\cup M,P}_{CC} SC_1' + SC_2'}$$

$$(CR_7)\quad \frac{C_1 \xrightarrow{l,\alpha^-,L,P}_{CC} C_1' \quad C_2 \xrightarrow{M}_{I} C_2' \quad l \notin Id(C_2)}{C_1 \mid C_2 \xrightarrow{l,\alpha^-,L\cup M,P}_{CC} C_1' \mid C_2'}$$

$$(CR_8)\quad \frac{C_1 \xrightarrow{l,\alpha^-,\emptyset,P_1}_{CC} C_1' \qquad C_2 \xrightarrow{l,\overline{\alpha}^-,\emptyset,P_2}_{CC} C_2'}{C_1 \mid C_2 \xrightarrow{l,\tau^-,\emptyset,\emptyset}_{CP} C_1' \mid C_2' \mid P_1 \mid P_2}$$

Fig. 4. The completion relation for conservative actions $\rightarrow_{CC} \subseteq \mathcal{C} \times \Theta_{CC}^- \times \mathcal{C}$.

$$(CR_9)\quad \frac{C_1 \xrightarrow{l,\alpha^-,L,P}_{CC} C_1' \qquad C_2 \xrightarrow{l,\overline{\alpha}^-,M_1,\emptyset}_{CP} C_2' \quad L = Id(C_1) \cap M_1}{C_1 \mid C_2 \xrightarrow{l,\tau^-,M_1\setminus L,\emptyset}_{CP} C_1' \mid C_2' \mid P}$$

Fig. 5. The completion relation for hybrid actions obtained by means of the other completion relations.

is that we have to take care that a summation configuration SC terminating a conservative action α could be composed in parallel with another summation configuration SC_2 terminating the action $\overline{\alpha}$ coupled with α. Now, if $\overline{\alpha}$ is preemptive, there may be some running actions β in SC_2 that should be interrupted, which implies that if there is an action $\overline{\beta}$ in SC_1 coupled with β, also $\overline{\beta}$ must be interrupted. For this reason, such a $\overline{\beta}$ must appear in L. Of course, if also $\overline{\alpha}$ is conservative, then in the transition by SC_1 used to infer the transition of $SC_1 \mid SC_2$, L will be empty.

The rules presented in Fig. 4 are at the basis of the definition of relation $\rightarrow_{CC}$. We implicitly assume rules symmetric to (CR_6) and (CR_7).

Rule (CR_5) deals with termination of a conservative action α in configuration $[\alpha]^l : (S,P)$. The process continues as S and, as expected by a conservative action, it produces a new copy of P, which, because of the inductive approach of the SOS semantics, cannot appear at the same syntactic level of the summation configuration S. Specifically, the process P will have to appear at the level of a parallel composition. To forward P at the correct syntactic level, P is exhibited as a label of the transition. The empty set used in the label denotes that no action is interrupted by completion of α.

Rule (CR_6) clearly justifies the terminology "conservative". When a conservative action α completes in a summation configuration SC_1 being part of a configuration $SC_1 + SC_2$, it is required neither that actions in SC_2 are interrupted, nor that SC_2 is canceled. This is clearly different from what happens when a preemptive action is completed (rule (CR_2)). More precisely, the set of actions interrupted in SC_2, namely M, is a subset of $Id(SC_2)$ since here not all the actions in SC_2 have to be interrupted. Furthermore, the new process created by the action, namely P in the rule, is exhibited as a transition label, since also in this case we are not yet at the syntactic level of a parallel composition.

Rule (CR_7) describes the case in which C_1 completes a conservative action α and the coupled action $\overline{\alpha}$ is not in C_2, since it runs in some other process running in parallel with $C_1 \mid C_2$.

Rule (CR_8) deals with the completion of two coupled conservative actions. As expected, the system exhibits an internal action τ, the label shows that no action has to be interrupted, and both the processes appearing in the labels of the transitions of both configurations C_1 and C_2, namely P_1 and P_2, are put in parallel with C_1' and C_2'. Notice that this last derivation is a derivation for $\rightarrow_{CP}$ rather than $\rightarrow_{CC}$. The reason for this is that P_1 and P_2 are already at the correct syntactic level and do not require to be lifted anymore.

Completion of both conservative and preemptive actions

We have to deal with the completion of two coupled actions α and $\overline{\alpha}$ such that one of them is conservative and the other preemptive. To this purpose, we add the rule shown in Fig. 5 and we implicitly assume a symmetric rule.

Notice that $L = Id(C_1) \cap M_1$ expresses that the actions running in C_1 performing the conservative actions and that have to be interrupted are those that were coupled with actions running in C_2 performing the preemptive action. In fact, such a coupling is the only reason we have to interrupt actions running in C_1.

A toy example

In order to show how our computational semantics works, we discuss the modeling of a toy multiscale system where we consider two populations. At a higher level of abstraction we consider a cell C, and at a lower level a generic protein P. A cell C can be involved in a process leading to its duplication. Also, it can be involved in some low-level reactions (i.e. DNA

transcription inside its nucleus) leading to the creation, in the environment outside C, of a protein of species P. Of course, we consider this model at a level of detail such that we do not need to take into account any other possible population of either cells or proteins which could be involved in the dynamics.

In the context of chemically reacting systems such a system may be described by two populations C and P, and by two reactions R_1 and R_2 such that

$$R_1 : \ C \to C + C \qquad R_2 : \ C \to C + P.$$

Reactions R_1 and R_2 model the non linear growth of cell C and the production of a protein P by a cell C, respectively. Notice that, at this level of detail, the production of protein P depends on the cell C where all the details of the biological process leading to the creation of the protein are abstracted away. The initial state of the system can be defined to have a precise initial number of cells C and proteins P in the environment.

We model now such a system in PAPC, and we show how the SOS semantics models the behavior of the populations. As PAPC is based on the paradigm processes-as-molecules, we start by assuming two types of processes for each species which, for clarity, are named C and P. As we want to model two reactions, we assume the following set of actions $\{\alpha, \overline{\alpha}, \gamma, \overline{\gamma}\}$ where α (resp. $\overline{\alpha}$) and γ (resp. $\overline{\gamma}$) model reaction R_1 and R_2, respectively. Also, as in PAPC the communication is dyadic and the reactions use a single reactant, we define two auxiliary process X^α and X^γ, used to model the communications on α and γ, respectively.

Reaction R_1 creates two new different cells able to start again, if possible, the duplication process. In the context of PAPC we model R_1 by using preemptive actions for both α and $\overline{\alpha}$ since the duplication of a cell interrupts all the low-level protein-transcription event inside the duplicated cell. Differently, the actions modeling reaction R_2 are conservative since the protein-transcription event does not interrupt the duplication process started by a cell.

The PAPC processes are defined as

$$C \stackrel{\text{def}}{=} \alpha.(C \mid C) + \gamma : (G, P) \qquad G \stackrel{\text{def}}{=} \gamma : (G, P) \qquad X^\alpha \stackrel{\text{def}}{=} \overline{\alpha}.(X^\alpha \mid X^\alpha) \qquad X^\gamma \stackrel{\text{def}}{=} \overline{\gamma} : (X^\gamma, \mathbf{0}).$$

Notice that we do not give a definition of the process P since it appears only as a product of the events we want to model, and hence we are not interested in the interactions it may have in the system.

As expected, process C can perform two actions. Action α produces the two new copies of C. Such an action is performed by synchronizing with the auxiliary process X^α which will produce two copies of itself to permit the duplication of the new cells. Indeed, the number of copies of the auxiliary process X^α must grow with the same law of growth for the cells C. Both C and X^α behave as preemptive in α and $\overline{\alpha}$, as expected. Process C can also perform, by synchronizing with X^γ, the action γ. The result of such action is to produce a new protein P without interrupting its running duplication, if any. The number of processes X^γ in the system bounds the number of cells which can simultaneously produce a protein P.

We discuss now some features of the semantics of PAPC for a simple system S described by the process

$$S = C \mid X^\alpha \mid X^\gamma.$$

In S both the reactions may fire. We assume reaction R_1 to fire first. To this extent, the semantics permits us to observe handshaking derivations as

$$C \xrightarrow{1,\alpha^+}_H [\alpha]^1.(C \mid C) + \gamma : (G, P) \qquad X^\alpha \xrightarrow{1,\overline{\alpha}^+}_H [\overline{\alpha}]^1.(X^\alpha \mid X^\alpha).$$

Then the whole process S performs a derivation as

$$S \xrightarrow{1,\tau^+}_H [\alpha]^1.(C \mid C) + \gamma : (G, P) \mid [\overline{\alpha}]^1.(X^\alpha \mid X^\alpha) \mid X^\gamma = S'$$

where the new process S' is such that the action α is now running in C and in X^α, with identifier 1. In S' action α cannot start, but just complete. We assume reaction R_2 fires next. The semantics permits us to observe the handshaking derivations

$$\gamma : (G, P) \xrightarrow{1,\gamma^+}_H [\gamma]^1 : (G, P) \qquad X^\gamma \xrightarrow{1,\overline{\gamma}^+}_H [\overline{\gamma}]^1 : (X^\gamma, \mathbf{0}).$$

The composition of these derivations resolves the conflicts of the colliding identifiers such that the derivation for C will be

$$[\alpha]^1.(C \mid C) + \gamma : (G, P) \xrightarrow{2,\gamma^+}_H [\alpha]^1.(C \mid C) + [\gamma]^2 : (G, P)$$

and the whole system performs the following derivation

$$S' \xrightarrow{2,\tau^+}_H [\alpha]^1.(C \mid C) + [\gamma]^2 : (G, P) \mid [\overline{\alpha}]^1.(X^\alpha \mid X^\alpha) \mid [\overline{\gamma}]^2 : (X^\gamma, \mathbf{0}) = S''$$

where in S'' all the possible actions are running. We consider now two different cases: (a) R_1 completes before R_2 and (b) vice versa.

(a) Reaction R_1 completes before R_2: in this case actions α and $\overline{\alpha}$ complete before action γ and $\overline{\gamma}$, interrupting them. The semantics permits us to derive transitions as

$$[\alpha]^1.(C \mid C) \xrightarrow{1,\alpha^-,\emptyset,\emptyset}_{CP} C \mid C \qquad [\overline{\alpha}]^1.(X^\alpha \mid X^\alpha) \xrightarrow{1,\overline{\alpha}^-,\emptyset,\emptyset}_{CP} X^\alpha \mid X^\alpha.$$

Actions in C and in X^γ have to be interrupted, hence we derive

$$[\alpha]^1.(C \mid C) + [\gamma]^2 : (G, P) \xrightarrow{1,\alpha^-,\{2\},\emptyset}_{CP} C \mid C \qquad [\overline{\gamma}]^2 : (X^\gamma, \mathbf{0}) \xrightarrow{\{2\}}_{I} X^\gamma$$

where $\{2\}$ denotes the actions to be interrupted. Consequently, the whole process S'' will perform the transition

$$S'' \xrightarrow{1,\tau^-,\emptyset,\emptyset}_{CP} C \mid C \mid X^\alpha \mid X^\alpha \mid X^\gamma$$

where in the resulting process no actions are running, as expected, and there are two cells and two auxiliary processes X^α.

(b) Reaction R_2 completes before R_1: in this case actions γ and $\overline{\gamma}$ complete before actions α and $\overline{\alpha}$. The semantics permits to derive transitions as

$$[\overline{\gamma}]^2 : (X^\gamma, \mathbf{0}) \xrightarrow{2,\overline{\gamma}^-,\emptyset,\mathbf{0}}_{CC} \mathbf{0} \qquad [\gamma]^2 : (G, P) \xrightarrow{2,\gamma^-,\emptyset,P}_{CC} G.$$

No actions have to be interrupted in any process, hence we derive

$$[\alpha]^1.(C \mid C) \xrightarrow{\emptyset}_{I} [\alpha]^1.(C \mid C) \qquad [\alpha]^1.(C \mid C) + [\gamma]^2 : (G, P) \xrightarrow{2,\gamma^-,\emptyset,\mathbf{0}}_{CC} [\alpha]^1.(C \mid C) + G.$$

The whole process S'' will then perform the transition

$$S'' \xrightarrow{2,\tau^-,\emptyset,\emptyset}_{CP} [\alpha]^1.(C \mid C) + G \mid [\overline{\alpha}]^1.(X^\alpha \mid X^\alpha) \mid X^\gamma \mid P \mid \mathbf{0}$$

where, as expected, in the resulting process only one action is still running (cell division) and a single protein P has been produced.

3.2. Operational correspondence

In this section we prove a result of operational correspondence between the reduction semantics defined in Section 2.2 and the compositional semantics defined in Section 3.1. In particular, we prove that transitions $\xrightarrow{l+}$ and $\xrightarrow{l-}$ in the reduction semantics have corresponding transitions with τ^+ and τ^- in their labels, respectively, in the compositional semantics, and vice versa.

We start by giving some auxiliary lemmas. The first one states a property of identifiers used in the handshaking relation of the compositional semantics.

Lemma 3. *Given a PAPC process configuration $C \in \mathcal{C}$, it holds:*

$$C \xrightarrow{l,\alpha^+}_{H} C' \text{ with } \alpha \in Act_\tau \text{ implies } l = \min(\mathbb{N} \setminus Id(C)).$$

Proof. Trivial induction on the derivation of $C \xrightarrow{l,\alpha^+}_{H} C'$. □

The second lemma states that structurally congruent process configurations perform the same transitions in the compositional semantics.

Lemma 4. *Given two PAPC process configurations $C_1, C_2 \in \mathcal{C}$ such that $C_1 \equiv C_2$, it holds:*

$$C_1 \xrightarrow{\ell}_{r} C_1' \qquad \text{if and only if} \qquad C_2 \xrightarrow{\ell}_{r} C_2'$$

with $C_1' \equiv C_2'$ and $r \in \{H, I, CP, CC\}$.

Proof. By cases on the axioms in Definition 3:

- (commutativity of $+$) follows from the fact that rules (H_3), (H_4), (CR_2) and (CR_6) are assumed to have analogous symmetric rules and from commutativity of $\cup$ in rules (I_9) and (I_{10});
- ($\mathbf{0}$ as neutral element of $+$) by rules (H_3), (I_9), (CR_2) and (CR_6) we have that $S + \mathbf{0}$ performs the same transitions as S (other rules for $+$ are not applicable to $S + \mathbf{0}$);
- (associativity of $+$) by rules (H_3) and (H_4) we have that if $(SC_1 + SC_2) + SC_3 \xrightarrow{l,\alpha^+}_{H} SC'$, then $SC' = (SC_1' + SC_2') + SC_3'$ and there exists $i \in \{1, 2, 3\}$ such that
  1. $SC_i \xrightarrow{l,\alpha^+}_{H} SC_i''$, where $SC_i' = SC_i''[l'/l]$ with, by Lemma 3, $l' = \min(\mathbb{N} \setminus Id((SC_1 + SC_2) + SC_3))$;
  2. $SC_j' = SC_j$ if $j \in \{1, 2, 3\}$ and $i \neq j$

But $SC_i \xrightarrow{l,\alpha^+}_H SC_i''$ implies $SC_1 + (SC_2 + SC_3) \xrightarrow{l,\alpha^+}_H SC_1' + (SC_2' + SC_3')$ (by applying rules (H_3) and (H_4)), and it holds $SC_1' + (SC_2' + SC_3') \equiv SC'$. The same holds for transitions obtained by applying rules (I_9), (CR_2) and (CR_6);

- (recursive definitions) follows immediately from the definition of rule (H_8);
- (commutativity of $|$) follows from the fact that rules (H_5), (H_6), (CR_3), (CR_7) and (CR_9) are assumed to have analogous symmetric rules, from the fact that rule transitions derived by applying rules (H_7), (I_{10}) and (CR_8) are the same for $C_1 \mid C_2$ and for $C_2 \mid C_1$, and from commutativity of $\cup$ and $\cap$ in rules (I_{10}), (CR_3) and (CR_4);
- ($\mathbf{0}$ as neutral element of $|$) by rules (H_5), (I_{10}), (CR_3) and (CR_7) we have that $C \mid \mathbf{0}$ performs the same transitions as C (other rules for $|$ are not applicable to $C \mid \mathbf{0}$);
- (associativity of $|$) the cases of transitions in which only one of the three components C_1, C_2 and C_3 is changing is analogous to the case of the associativity of $+$, but by using rules (H_5), (I_{10}), (CR_3) and (CR_7). The cases of transitions describing synchronizations, and obtained by applying rules (H_7), (CR_4), (CR_8) and (CR_9) is still similar, but requires that two of the three components change. □

The next two lemmas relate the interruption and the completion relations of the compositional semantics with the *Reset* function used in the reduction semantics.

Lemma 5. *Given any PAPC process configuration $C \in \mathcal{C}$ and any $L \subseteq Id(C)$ it holds:*

$$C \xrightarrow{L}_I C' \qquad \text{if and only if} \qquad C' = Reset(C, L).$$

Proof. We have to prove two implications: $C \xrightarrow{L}_I C'$ implies $C' = Reset(C, L)$ and $C' = Reset(C, L)$ implies $C \xrightarrow{L}_I C'$. The former implication can be proved by trivial induction on the derivation of $C \xrightarrow{L}_I C'$, and the latter by trivial structural induction on C. □

Lemma 6. *Given any PAPC summation configuration $SC \in \mathcal{SC}$ it holds:*

$$[\alpha]^l : (S, P) + SC \xrightarrow{l,\alpha^-,L,P}_{CC} S + SC' \quad \text{if and only if} \quad L = Id(SC) \text{ and } SC' = Reset(SC, L).$$

Proof. We have to prove two implications: the first one is $[\alpha]^l : (S, P) + SC \xrightarrow{l,\alpha^-,L,P}_{CC} S + SC'$ implies $L = Id(SC)$ and $C' = Reset(C, L)$; the second one is $L = Id(SC)$ and $C' = Reset(C, L)$ imply $[\alpha]^l : (S, P) + SC \xrightarrow{l,\alpha^-,L,P}_{CC} S + SC'$. The first implication can be proved by induction on the derivation of $[\alpha]^l : (S, P) + SC \xrightarrow{l,\alpha^-,L,P}_{CC} S + SC'$ by using Lemma 5 in the inductive case. The second implication can be proved by structural induction on SC, again by using Lemma 5 in the inductive case. □

Finally, we give the main theorem on operational correspondence between the two considered semantics.

Theorem 1. *Given any PAPC process configuration $C \in \mathcal{C}$, the following relationships hold:*

1. $C \xrightarrow{l+} C'$ *if and only if* $C \xrightarrow{l,\tau^+}_H C'$;
2. $C \xrightarrow{l-} C'$ *if and only if* $C \xrightarrow{l,\tau^-,\emptyset,\emptyset}_{CP} C'$.

Proof. We have to prove $C \xrightarrow{l+} C'$ implies $C \xrightarrow{l,\tau^+}_H C'$, $C \xrightarrow{l,\tau^+}_H C'$ implies $C \xrightarrow{l+} C'$, $C \xrightarrow{l-} C'$ implies $C \xrightarrow{l,\tau^-,\emptyset,\emptyset}_{CP} C'$ and $C \xrightarrow{l,\tau^-,\emptyset,\emptyset}_{CP} C'$ implies $C \xrightarrow{l-} C'$:

- ($C \xrightarrow{l+} C'$ implies $C \xrightarrow{l,\tau^+}_H C'$) Let us assume that transition $C \xrightarrow{l+} C'$ has been derived by applying rule $(+2)$ of the reduction semantics (the cases of $(+1)$ and $(+3)$ are analogous). Lemma 4 and the closure of the reduction semantics with respect to $\equiv$ allow us to assume that $C = (\alpha.P_1 + SC_1 \mid \overline{\alpha} : (S_2, P_2) + SC_2) \mid C_1$ and $C' = ([\alpha]^l.P_1 + SC_1 \mid [\overline{\alpha}]^l : (S_2, P_2) + SC_2) \mid C_1$. In this case, a transition $C \xrightarrow{l,\tau^+}_H C'$ can be obtained by the following derivation:

$$(H_{5/6})\ \frac{(H_7)\ \dfrac{(H_{3/4})\ \dfrac{(H_1)}{\alpha.P_1 \xrightarrow{1,\alpha^+}_H [\alpha]^1.P_1} \qquad (H_{3/4})\ \dfrac{(H_1)}{\overline{\alpha}:(S_2,P_2) \xrightarrow{1,\overline{\alpha}^+}_H [\overline{\alpha}]^1_{(S_2,P_2)}}}{\alpha.P_1+SC_1 \xrightarrow{l_1,\alpha^+}_H [\alpha]^l 1.P_1+SC_1 \qquad \overline{\alpha}:(S_2,P_2)+SC_2 \xrightarrow{l_2,\overline{\alpha}^+}_H [\overline{\alpha}]^{l_1}_{(S_2,P_2)}+SC_2}}{\alpha.P_1+SC_1|\overline{\alpha}:(S_2,P_2)+SC_2 \xrightarrow{l',\tau^+}_H [\alpha]^{l'}.P_1+SC_1|[\overline{\alpha}]^{l'}_{(S_2,P_2)}+SC_2}$$

$$\frac{}{(\alpha.P_1 + SC_1 \mid \overline{\alpha} : (S_2, P_2) + SC_2) \mid C_1 \xrightarrow{l,\tau^+}_H ([\alpha]^l.P_1 + SC_1 \mid [\overline{\alpha}]^l : (S_2, P_2) + SC_2) \mid C_1}$$

where the choice between (H_3) and (H_4), and between (H_5) and (H_6) depends on the already used identifiers in the portion of configuration they are applied to. Lemma 3 ensures that $l = min((N) \setminus Id(SC_1) \cup Id(SC_2) \cup Id(C_1))$;

- ($C \xrightarrow{l,\tau^+}_H C'$ implies $C \xrightarrow{l+} C'$) The only rule that introduces τ in the label of a handshaking transition is (H_7), hence any derivation of $C \xrightarrow{l,\tau^+}_H C'$ has an application of such a rule in its derivation tree. The premise of (H_7) requires C to contain two components C_1 and C_2 such that $C_1 \xrightarrow{l_1,\alpha^+}_H C_1'$ and $C_2 \xrightarrow{l_2,\overline{\alpha}^+}_H C_2'$, with C having the form $C_1|C_2$ and C' the form $C_1'|C_2'$. Since the only rules introducing α and $\overline{\alpha}$ are (H_1) and (H_2), we have that such two instances of these rules are used in any derivation tree of $C \xrightarrow{l,\tau^+}_H C'$ (actually, either only (H_1) twice, or only (H_2) twice, or both (H_1) and (H_2) once). These considerations on the derivation trees of $C \xrightarrow{l,\tau^+}_H C'$ allow us to conclude that there must be two components of C which are composed in parallel and which are ready to execute α and $\overline{\alpha}$, respectively. Let us assume that one of these actions is used as a preemptive action, namely as $\alpha.P_1$, and the other as a conservative action, namely as $\overline{\alpha} : (S_2, P_2)$. By Lemma 4 we can assume, without loss of generality, that $C = (\alpha.P_1 + SC_1 \mid \overline{\alpha} : (S_2, P_2) + SC_2) \mid C_3$, and hence it is easy to verify that $C' = ([\alpha]^l.P_1 + SC_1 \mid [\overline{\alpha}]^l : (S_2, P_2) + SC_2) \mid C_3$. Lemma 3 ensures that $l = min((N) \setminus Id(SC_1) \cup Id(SC_2) \cup Id(C_3))$ and from this it follows that by applying rule ($+2$) of the reduction semantics we can derive $C \xrightarrow{l+} C'$. The cases in which α and $\overline{\alpha}$ are not both used as preemptive actions are analogous, and involve rules ($+1$) and ($+3$) of the reduction semantics;
- ($C \xrightarrow{l-} C'$ implies $C \xrightarrow{l,\tau^-,\emptyset,\emptyset}_{CP} C'$) Let us consider the case in which $C \xrightarrow{l-} C'$ has been derived by applying rule (-2) of the reduction semantics. Lemma 4 and the closure of the reduction semantics with respect to $\equiv$ allow us to assume that $C = ([\alpha]^l.P_1 + SC_1 \mid [\overline{\alpha}]^l : (S_2, P_2) + SC_2) \mid C_1$ and $C' = (P_1 \mid S_2 + SC_2' \mid P_2) \mid C_1'$.

  By the premise of rule (-2) we know that $C_1' = Reset(C_1, L)$ and $SC_2' = Reset(SC_2, L)$ where $L = Id(SC_1)$. The set L is not necessarily a subset of $Id(C_1)$ since there may be some incomplete synchronizations between SC_1 and SC_2 (that necessarily do not involve C_1). Consequently, $Id(SC_1) \cap Id(SC_2)$ might be not empty. Let us consider $L' = L \cap Id(C_1)$. By Lemma 1 we know that $L' = L \setminus Id(SC_2)$. Moreover, by Lemma 2 it follows that $C_1' = Reset(C_1, L) = Reset(C_1, L')$. Now, since $L' \subseteq Id(C_1)$, we can exploit Lemma 5 and obtain $C_1 \xrightarrow{L'}_I C_1'$. In addition, we can exploit Lemma 6 and obtain $[\overline{\alpha}]^l : (S_2, P_2) + SC_2 \xrightarrow{l,\overline{\alpha}^-,Id(SC_2),P_2}_{CC} S_2 + SC_2$.

  The obtained transitions can be used in the following derivation tree for $C \xrightarrow{l,\tau^-,\emptyset,C'}_{CP}$:

$$(CR_3)\ \frac{(CR_9)\ \dfrac{(CR_1) \text{ and } (CR_2)\ \vdots}{[\alpha]^l.P_1+SC_1 \xrightarrow{l,\alpha^-,Id(SC_1),\emptyset}_{CP} P_1} \qquad [\overline{\alpha}]^l:(S_2,P_2)+SC_2 \xrightarrow{l,\overline{\alpha}^-,Id(SC_2),P_2}_{CC} S_2+SC_2' \qquad C_1 \xrightarrow{L'}_I C_1'}{([\alpha]^l.P_1 + SC_1 \mid [\overline{\alpha}]^l : (S_2, P_2) + SC_2) \mid C_1 \xrightarrow{l,\tau^-,\emptyset,\emptyset}_{CP} (P_1 \mid S_2 + SC_2' \mid P_2) \mid C_1'}$$

  The cases in which $C \xrightarrow{l-} C'$ is derived by applying either rule (-1) or rule (-3) are similar.
- ($C \xrightarrow{l,\tau^-,\emptyset,\emptyset}_{CP} C'$ implies $C \xrightarrow{l-} C'$) There are three rules that introduce τ in the label of a completion transition, namely (CR_4), (CR_8) and (CR_9). Hence, any derivation of $C \xrightarrow{l,\tau^-,\emptyset,\emptyset}_{CP} C'$ has an application of one of these three rules in its derivation tree. Let us consider the case in which rule (CR_9) is applied. The premise of such a rule requires C to contain two components C_1 and C_2 such that $C_1 \xrightarrow{l,\alpha^-,L,P}_{CC} C_1'$ and $C_2 \xrightarrow{l,\overline{\alpha}^-,M_1,\emptyset}_{CP} C_2'$.

  It is immediate to see that any derivation tree for $C_1 \xrightarrow{l,\alpha^-,L,P}_{CC} C_1'$ must consist of an application of rule (CR_5) followed by a (possibly empty) sequence of applications of (CR_6) and then by a (possibly empty) sequence of applications of (CR_7). As a consequence, by Lemma 4 we can assume $C_1 = [\alpha]^l : (S_1, P) + SC_1 \mid C_3$ for some S_1, SC_1 and C_3. By definition of (CR_5) and (CR_6) we have $L \subseteq Id(SC_1) \cup Id(C_3)$.

  Similarly, any derivation tree for $C_2 \xrightarrow{l,\overline{\alpha}^-,M_1,\emptyset}_{CP} C_2'$ must consist of an application of rule (CR_1) followed by a (possibly empty) sequence of applications of (CR_2) and then by a (possibly empty) sequence of applications of (CR_3). As a consequence, by Lemma 4 we can assume $C_2 = [\overline{\alpha}]^l.P_2 + SC_2 \mid C_4$ for some P_1, SC_2 and C_4. By definition of (CR_2) and (CR_3) we have $M_1 = Id(SC_2) \setminus M$ with $M \subseteq Id(C_4)$.

  Up to now we have understood that C contains a subterm $C_1 \mid C_2 = [\alpha]^l : (S_1, P_1) + SC_1 \mid C_3 \mid [\overline{\alpha}]^l.P_2 + SC_2 \mid C_4$ which is the subject of application of rule (CR_9). By Lemma 4 we can use structural congruence to rearrange the subterm as follows: $C_1 \mid C_2 \equiv [\alpha]^l : (S_1, P_1) + SC_1 \mid [\overline{\alpha}]^l.P_2 + SC_2 \mid C_3 \mid C_4$. In this way we can assume that (CR_9) is used to derive a transition for the simpler subterm $[\alpha]^l : (S_1, P_1) + SC_1 \mid [\overline{\alpha}]^l.P_2 + SC_2$. Consequently, we can assume C_1 and C_2 to be $[\alpha]^l : (S_1, P_1) + SC_1$ and $[\overline{\alpha}]^l.P_2 + SC_2$, respectively, and we have $L \subseteq Id(SC_1)$ and $M_1 = Id(SC_2)$.

  The premise of (CR_9) requires $L = Id(C_1) \cap M_1$, that is $L = Id(C_1) \cap Id(SC_2) = Id(SC_1) \cap Id(SC_2)$. Hence, by applying ($CR_9$) we obtain $C_1 \mid C_2 \xrightarrow{l,\tau^-,Id(SC_2)\setminus(Id(SC_1)\cap Id(SC_2)),\emptyset}_{CP} C_1' \mid C_2' \mid P$, that is $C_1 \mid C_2 \xrightarrow{l,\tau^-,Id(SC_2)\setminus Id(SC_1),\emptyset}_{CP} C_1' \mid C_2' \mid P$.

Moreover, by the transition of C_1 in the premise of rule (CR_9) and by Lemmas 2 and 6 we can conclude that $C_1' = S_1 + SC_1'$ with $SC_1' = Reset(SC_1, Id(SC_1) \cap Id(SC_2)) = Reset(SC_1, Id(SC_2))$. Now, in order to derive a transition for the whole process configuration C we can observe that $C = (C_1 \mid C_2) \mid \overline{C}$, for some $\overline{C}$ that includes the C_3 and C_4 subterms we previously moved out of the subject of application (CR_9). Hence, rule (CR_3) can be applied with the transition performed by $C_1 \mid C_2$ as a premise. Since the transition performed by C is $C \xrightarrow{l,\tau^-,\emptyset,\emptyset}_{CP} C'$ we must have, in the premise of (CR_3), that $\overline{C} \xrightarrow{Id(SC_2)\setminus Id(SC_1)}_{I} \overline{C'}$, and by Lemma 5 we know that $\overline{C'} = Reset(\overline{C}, Id(SC_2) \setminus Id(SC_1))$. Finally, by Lemma 2, we have that also $\overline{C'} = Reset(\overline{C}, Id(SC_2))$ holds, hence rule (-2) of the reduction semantics can be applied to derive $C \xrightarrow{l-} C'$.

The case in which rule (CR_4) is applied in place of (CR_9) differs with respect to the case of (CR_9) in two things: (i) the rules used to derive the transition of C_1 are (CR_1) and (CR_2) rather than (CR_5) and (CR_6), and (ii) the set of identifiers appearing in the labels of the transitions of C_1 and C_2 are subject to more complex constraints than in the case of (CR_9). As regards (ii), by Lemma 4 we can assume C_1 and C_2 to be $[\alpha]^l.P_1 + SC_1$ and $[\overline{\alpha}]^l.P_2 + SC_2$, respectively (similarly as we did in the case of (CR_9)). This means that in the premise of (CR_4) we can assume both L_2 and M_2 to be empty (this follows immediately from the definitions of rules (CR_2) and (CR_3)). Consequently, the conclusion of rule (CR_4) becomes $C_1 \mid C_2 \xrightarrow{l,\tau^-,(L_1\setminus M_1)\cup(M_1\setminus L_1),\emptyset}_{CP} C_1' \mid C_2'$. In addition, from the definition of (CR_2) we can conclude that $L_1 = Id(SC_1)$ and $M_1 = Id(SC_2)$. This means that the conclusion of (CR_4) corresponds to $C_1 \mid C_2 \xrightarrow{l,\tau^-,(Id(SC_1)\setminus Id(SC_2))\cup(Id(SC_2)\setminus Id(SC_1)),\emptyset}_{CP} C_1' \mid C_2'$ that is, by Lemma 1, $C_1 \mid C_2 \xrightarrow{l,\tau^-,(Id(SC_1)\cap Id(\overline{C}))\cup(Id(SC_2)\cap Id(\overline{C})),\emptyset}_{CP} C_1' \mid C_2'$, namely $C_1 \mid C_2 \xrightarrow{l,\tau^-,(Id(SC_1)\cup Id(SC_2))\cap Id(\overline{C}),\emptyset}_{CP} C_1' \mid C_2'$ (where $\overline{C}$ is as in the case of (CR_9)). We can conclude that rule (-1) of the reduction semantics can be applied to derive $C \xrightarrow{l-} C'$ after an argument on the transition performed by $\overline{C}$ and on $\overline{C'}$ analogous to that given in the case of (CR_9).

The cases in which rule (CR_8) is applied is analogous to those of (CR_9) and (CR_4). □

From the correspondence between the reduction and the compositional semantics we can easily obtain the following result of preservation of well-formedness and completeness of process configurations.

Corollary 1. *Given a well-formed and complete process configuration $C \in \mathcal{C}$, if either $C \xrightarrow{l,\tau^+}_{H} C'$ or $C \xrightarrow{l,\tau^-,\emptyset,\emptyset}_{CP} C'$, then C' is well-formed and complete as well.*

Proof. Follows immediately from Theorem 1 and Proposition 1. □

In addition we can prove that in the compositional semantics it is possible to derive from a well-formed and complete process configuration only either a handshaking transition or a completion transition of the $\rightarrow_{CP}$ relation with τ and empty sets of action identifiers in the label.

Proposition 3. *Given a well-formed and complete process configuration $C \in \mathcal{C}$, there does not exist any $C' \in \mathcal{C}$ such that*

- *either $C \xrightarrow{l,\alpha^-,L,P}_{CC} C'$,*
- *or $C \xrightarrow{l,\alpha^-,L_1,L_2}_{CP} C'$ with $\alpha \neq \tau$ or $L_1 \cup L_2 \neq \emptyset$.*

Proof. Let us assume that there exists C' such that $C \xrightarrow{l,\alpha^-,L,P}_{CC} C'$ can be derived. By definition of the completion relation for conservative actions we have $\alpha \neq \tau$. This means that somewhere in the derivation tree of the transition rule (CR_5) has been applied to a subterm of C containing a running action α with identifier l. By definition of well-formedness and completeness we have that C has to contain also the coupled running action $\overline{\alpha}$ with the same identifier. Moreover, such a coupled running action has to be in a subterm of C that is composed in parallel with the subterm containing α. This means that also rule (CR_7) is used in the derivation tree of the transition. But rule (CR_7) has a premise that makes it impossible for l to be in a subterm composed in parallel with the subterm executing α. This contradiction implies that there does not exist C' such that $C \xrightarrow{l,\alpha^-,L,P}_{CC} C'$ can be derived.

Now, let us assume that there exists C' such that $C \xrightarrow{l,\alpha^-,L_1,L_2}_{CP} C'$ can be derived with $\alpha \neq \tau$. Also in this case we reach a contradiction with a reasoning analogous to that of the previous case, but with considering rules (CR_1) and (CR_4) in place of (CR_7) and (CR_5).

Finally, let us assume that there exists C' such that $C \xrightarrow{l,\alpha^-,L_1,L_2}_{CP} C'$ can be derived with $L_1 \cup L_2 \neq \emptyset$. By the proof of the previous case it follows that $\alpha = \tau$, hence there exist α' and $\overline{\alpha}'$ that were coupled running action in C and that complete their execution in the described step. In this case we have that in the derivation tree of the transition rule (CR_2) has been applied, possibly several times, to the two summation configurations containing α' and $\overline{\alpha}'$. Since C is well-formed and complete, we have that all of the running action coupled with the running actions in the two considered summation configurations must be in C as well. Condition $(L_1 \cap Id(C_2)) \subseteq M$ in the premise of rule (CR_3) and conditions $(M_1 \cap Id(C_1)) \subseteq (L_1 \cup L_2)$ and $(L_1 \cap Id(C_2)) \subseteq (M_1 \cup M_2)$ in the premise of rule (CR_4) ensure that all of the necessary running actions have been interrupted. Moreover, the definition of the sets of action identifiers in the conclusions of rules (CR_3) and (CR_4) is such that all of the interrupted running actions are removed. This leads to a contradiction as we assumed $L_1 \cup L_2 \neq \emptyset$. □

This result has two consequences. The first is that both the completion relations for preemptive and conservative actions are well-defined, in the sense that it is not possible to derive a transition in which an action coupled with another that is interrupted is not reset. The second consequence is that the completion relation for conservative actions, namely $\rightarrow_{CC}$, turns out to be only an auxiliary relation used for compositionality reasons. Actually, the same holds for the interruption relation $\rightarrow_I$. Even if $\rightarrow_{CC}$ and $\rightarrow_I$ do not describe a step of a well-formed and complete configuration they are necessary to describe a step of a portion of a configuration, namely a well-formed and possibly not complete configuration. As a consequence these transition relations will be taken into account in the following in the definition of a bisimulation equivalence for PAPC.

3.3. Bisimulation equivalence for PAPC

Bisimulation equivalence is a central notion in concurrency theory. For processes with a higher order behavior, namely processes whose behavior is described by portions of transition systems in which processes appear in the labels, the notion of bisimulation is usually replaced by a higher order notion of bisimulation [3,15,50,41], which can be rephrased in our setting as follows:

Definition 7. A symmetric relation $\mathcal{R} \subseteq \mathcal{C} \times \mathcal{C}$ is a *bisimulation* if and only if whenever $(C_1, C_2) \in \mathcal{R}$, then it holds that:

- if $C_1 \xrightarrow{\ell}_r C_1'$ for any $C_1' \in \mathcal{C}$, $r \in \{H, I, CP\}$ and label ℓ, then $C_2 \xrightarrow{\ell}_r C_2'$ for some $C_2' \in \mathcal{C}$ such that $(C_1', C_2') \in \mathcal{R}$.
- if $C_1 \xrightarrow{l,\alpha^-,L,P_1}_{CC} C_1'$ for any $C_1' \in \mathcal{C}$, $l \in \mathbb{N}$, $\alpha \in Act$, $L \subseteq \mathbb{N}$ and $P_1 \in \mathcal{P}$, then $C_2 \xrightarrow{l,\alpha^-,L,P_2}_{CC} C_2'$ for some $C_2' \in \mathcal{C}$ and $P_2 \in \mathcal{P}$ such that $(C_1', C_2') \in \mathcal{R}$ and $(P_1, P_2) \in \mathcal{R}$.

The union of all bisimulations is, in turn, a bisimulation, which is denoted with $\approx$. For an algebraic treatment of bisimulation equivalence and to reason in a compositional way, a bisimulation is required to be a congruence.

Theorem 2. *Bisimulation is a congruence w.r.t. all PAPC operations.*

Proof. First of all let us recall that an SOS transition rule respects the *De Simone format* (see, e.g., [1]) if it is in the following form:

$$\frac{\{x_i \xrightarrow{\ell_i}_{Z_i} y_i \mid i \in I\}}{f(x_1, \ldots, x_n) \xrightarrow{\ell}_Z t}$$

where f is an arbitrary operator with n arguments, labels ℓ_i and ℓ are arbitrary, symbols Z_i and Z are arbitrary, and the following constraints are respected:

- $I \subseteq \{1, \ldots, n\}$;
- all x_i with $1 \leq i \leq n$ and y_i with $i \in I$ are distinct process variables and it holds that if $x_i = x_j$ for some $i, j \in I$ then $i = j$;
- t is a term built over the language syntax such that:
  - for all $i \in I$, no process y_i appears in t more than once, and no process x_i appears in t at all;
  - for all $i \in \{1, \ldots, n\} \setminus I$, no process x_i appears more than once in t;
  - no process variable except variables x_i and y_i appears in t.

It is well known (see, e.g., [1]) that if an SOS has only the classical recursion rule and transition rules in De Simone format, then bisimulation is a congruence. (Notice that side conditions in transition rules do not affect the result.) Since PAPC transition rules $(CR_5) - (CR_9)$ do not respect De Simone format, we cannot infer the thesis in an immediate way. What we do is that we prove the thesis in the standard way and we exploit the result by De Simone when this is possible.

First of all let $\mathcal{R}$ be the least congruence relation defined over PAPC configurations containing bisimulation. Formally, $\mathcal{R}$ is the least equivalence relation such that $\approx \subseteq \mathcal{R}$ and:

- $C_1 \ \mathcal{R} \ C_1'$ and $C_2 \ \mathcal{R} \ C_2'$ imply $C_1 \mid C_2 \ \mathcal{R} \ C_1' \mid C_2'$;
- $P \ \mathcal{R} \ P'$ implies $\alpha.P \ \ \mathcal{R} \ \ \alpha.P'$ and $[\alpha]^l.P \ \ \mathcal{R} \ \ [\alpha]^l.P'$;
- $S \ \mathcal{R} \ S'$ and $P \ \mathcal{R} \ P'$ imply $\alpha : (S, P) \ \ \mathcal{R} \ \ \alpha : (S', P')$ and $[\alpha]^l : (S, P) \ \ \mathcal{R} \ \ [\alpha]^l : (S', P')$;
- $SC_1 \ \mathcal{R} \ SC_1'$ and $SC_2 \ \mathcal{R} \ SC_2'$ imply $SC_1 + SC_2 \ \ \mathcal{R} \ \ SC_1' + SC_2'$.

Following the standard approach as in the literature, to prove the thesis it suffices to prove that bisimulation contains $\mathcal{R}$, so that one infers that bisimulation and $\mathcal{R}$, which is a congruence by definition, coincide. Hence we have to prove that given $C \ \mathcal{R} \ \hat{C}$ it holds that:

1. if $C \xrightarrow{(l,\alpha^+)}_H C'$ then $\hat{C} \xrightarrow{(l,\alpha^+)}_H \hat{C}'$ with $C' \ \mathcal{R} \ \hat{C}'$;
2. if $C \xrightarrow{L}_I C'$ then $\hat{C} \xrightarrow{L}_I \hat{C}'$ with $C' \ \mathcal{R} \ \hat{C}'$;

3. if $C \xrightarrow{(l,\alpha^-,L_1,L_2)}_{CP} C'$ then $\hat{C} \xrightarrow{(l,\alpha^-,L_1,L_2)}_{CP} \hat{C}'$ with $C' \mathcal{R} \hat{C}'$;
4. if $C \xrightarrow{(l,\alpha^-,L,P)}_{CC} C'$ then $\hat{C} \xrightarrow{(l,\alpha^-,L,\hat{P})}_{CC} \hat{C}'$ with $C' \mathcal{R} \hat{C}'$ and $P \mathcal{R} \hat{P}$.

As usual, we reason by induction over the depth of the derivation tree of the transition. The proof comes for free if the last transition rule used in the derivation tree is in De Simone format. So, it remains to prove the thesis when the last transition rule used in the derivation tree is one among $(CR_5) - (CR_9)$. Let us consider these five distinct cases:

- Case (CR_5). We have to consider the transition $C \xrightarrow{l,\alpha^-,\emptyset,P}_{CC} C'$ with $C \equiv [\alpha]^l : (S, P)$ and $C' \equiv S$. If $C \mathcal{R} \hat{C}$ is obtained from $C \approx \hat{C}$ then the thesis is immediate, otherwise it holds that $\hat{C} \equiv [\alpha]^l : (\hat{S}, \hat{P})$ with $S \mathcal{R} \hat{S}$ and $P \mathcal{R} \hat{P}$. By applying (CR_5) we infer $\hat{C} \xrightarrow{l,\alpha^-,\emptyset,\hat{P}}_{CC} \hat{S}$ and the thesis follows.
- Case (CR_6). We have to consider the transition $C \xrightarrow{l,\alpha^-,L\cup M,P}_{CC} C'$ inferred from $SC_1 \xrightarrow{l,\alpha^-,L,P}_{CC} SC_1'$ and $SC_2 \xrightarrow{M}_I SC_2'$, where $C \equiv SC_1 + SC_2$ and $C' \equiv SC_1' + SC_2'$. If $C \mathcal{R} \hat{C}$ is obtained from $C \approx \hat{C}$ then the thesis is immediate, otherwise it holds that $\hat{C} \equiv S\hat{C}_1 + S\hat{C}_2$ with $SC_1 \mathcal{R} S\hat{C}_1$ and $SC_2 \mathcal{R} S\hat{C}_2$. By the inductive hypothesis, it holds that $S\hat{C}_1 \xrightarrow{l,\alpha^-,L,\hat{P}}_{CC} S\hat{C}_1'$ and $S\hat{C}_2 \xrightarrow{M}_I S\hat{C}_2'$ with $SC_1' \mathcal{R} S\hat{C}_1'$, $SC_2' \mathcal{R} S\hat{C}_2'$ and $P \mathcal{R} \hat{P}$. By applying (CR_6) we infer $\hat{C} \xrightarrow{l,\alpha^-,L\cup M,\hat{P}}_{CC} S\hat{C}_1 + S\hat{C}_2$ and, since it is immediate that $C' \mathcal{R} S\hat{C}_1 + S\hat{C}_2$, the thesis follows.
- Case (CR_7). We have to consider the transition $C \xrightarrow{l,\alpha^-,L\cup M,P}_{CC} C'$ inferred from $C_1 \xrightarrow{l,\alpha^-,L,P}_{CC} C_1'$ and $C_2 \xrightarrow{M}_I C_2'$, where $C \equiv C_1 \mid C_2$ and $C' \equiv C_1' \mid C_2'$. If $C \mathcal{R} \hat{C}$ is obtained from $C \approx \hat{C}$ then the thesis is immediate, otherwise it holds that $\hat{C} \equiv \hat{C}_1 \mid \hat{C}_2$ with $C_1 \mathcal{R} \hat{C}_1$ and $C_2 \mathcal{R} \hat{C}_2$. By the inductive hypothesis, we infer that $\hat{C}_1 \xrightarrow{l,\alpha^-,L,\hat{P}}_{CC} \hat{C}_1'$ and $\hat{C}_2 \xrightarrow{M}_I \hat{C}_2'$ with $C_1' \mathcal{R} \hat{C}_1'$, $C_2' \mathcal{R} \hat{C}_2'$ and $P \mathcal{R} \hat{P}$. By applying (CR_7) we infer $\hat{C} \xrightarrow{l,\alpha^-,L\cup M,\hat{P}}_{CC} \hat{C}_1' \mid \hat{C}_2'$ and, since it is immediate that $C' \mathcal{R} \hat{C}_1' \mid \hat{C}_2'$, the thesis follows.
- Case (CR_8). We have to consider the transition $C \xrightarrow{l,\tau^-,\emptyset,\emptyset}_{CP} C'$ inferred from $C_1 \xrightarrow{l,\alpha^-,\emptyset,P_1}_{CC} C_1'$ and $C_2 \xrightarrow{l,\overline{\alpha}^-,\emptyset,P_2}_{CC} C_2'$, where $C \equiv C_1 \mid C_2$ and $C' \equiv C_1' \mid C_2' \mid P_1 \mid P_2$. If $C \mathcal{R} \hat{C}$ is obtained from $C \approx \hat{C}$ then the thesis is immediate, otherwise it holds that $\hat{C} \equiv \hat{C}_1 \mid \hat{C}_2$ with $C_1 \mathcal{R} \hat{C}_1$ and $C_2 \mathcal{R} \hat{C}_2$. By the inductive hypothesis, we infer that $\hat{C}_1 \xrightarrow{l,\alpha^-,\emptyset,\hat{P}_1}_{CC} \hat{C}_1'$ and $\hat{C}_2 \xrightarrow{l,\overline{\alpha}^-,\emptyset,\hat{P}_2}_{CC} \hat{C}_2'$ with $C_1' \mathcal{R} \hat{C}_1'$, $C_2' \mathcal{R} \hat{C}_2'$, $P_1 \mathcal{R} \hat{P}_1$ and $P_2 \mathcal{R} \hat{P}_2$. By applying (CR_8) we infer $\hat{C} \xrightarrow{l,\tau^-,\emptyset,\emptyset}_{CP} \hat{C}_1' \mid \hat{C}_2' \mid \hat{P}_1 \mid \hat{P}_2$, and, since it is immediate that $C_1' \mid C_2' \mid P_1 \mid P_2 \mathcal{R} \hat{C}_1' \mid \hat{C}_2' \mid \hat{P}_1 \mid \hat{P}_2$, the thesis follows.
- Case (CR_9). We have to consider the transition $C \xrightarrow{l,\tau^-,M_2\setminus L,\emptyset}_{CP} C'$ inferred from $C_1 \xrightarrow{l,\alpha^-,L,P}_{CC} C_1'$ and $C_2 \xrightarrow{l,\overline{\alpha}^-,M_2,\emptyset}_{CP} C_2'$, where $C \equiv C_1 \mid C_2$ and $C' \equiv C_1' \mid C_2' \mid P$. If $C \mathcal{R} \hat{C}$ is obtained from $C \approx \hat{C}$ then the thesis is immediate, otherwise it holds that $\hat{C} \equiv \hat{C}_1 \mid \hat{C}_2$ with $C_1 \mathcal{R} \hat{C}_1$ and $C_2 \mathcal{R} \hat{C}_2$. By the inductive hypothesis, we infer that $\hat{C}_1 \xrightarrow{l,\alpha^-,L,\hat{P}}_{CC} \hat{C}_1'$ and $\hat{C}_2 \xrightarrow{l,\overline{\alpha}^-,M_2,\emptyset}_{CP} \hat{C}_2'$ with $C_1' \mathcal{R} \hat{C}_1'$, $C_2' \mathcal{R} \hat{C}_2'$ and $P \mathcal{R} \hat{P}$. By applying (CR_9) we infer $\hat{C} \xrightarrow{l,\tau^-,M_2\setminus L,\emptyset}_{CP} \hat{C}_1' \mid \hat{C}_2' \mid \hat{P}$. Since it is immediate that $C_1' \mid C_2' \mid P \mathcal{R} \hat{C}_1' \mid \hat{C}_2' \mid \hat{P}$, the thesis follows. □

The bisimulation relation for PAPC is a very fine behavioral equivalence. This can be seen as a disadvantage, since with behavioral equivalences it is often desirable to be able to equate as many processes as possible. On the other hand, the fact that bisimulation turns out to be fine may have another meaning, namely that all the ingredients used in the process algebra play an important role. This does not happen, for instance, for the parallel composition in some variants of CCS where it can be reduced into an equivalent summation of processes.

We go through this last consideration via an example. Let us consider the following two PAPC processes

$$S_1 \stackrel{\text{def}}{=} \alpha.(S_1 \mid S_1) \qquad S_2 \stackrel{\text{def}}{=} \alpha : (S_2, S_2).$$

The behavior of the two processes is similar: both of them perform an action α and then continue as with two copies of the initial process. Even if the behavior of the two processes seems to be the same, it is immediate to see that the two processes are not bisimilar. In fact, they repeatedly perform the same handshaking transition $\xrightarrow{l,\alpha^+}_H$, but followed by two different completion transitions, namely $\xrightarrow{l,\alpha^-,\emptyset}_{CP}$ and $\xrightarrow{l,\alpha^-,\emptyset,S_2}_{CC}$, respectively. This permits to state that, in general, $S_1 \not\approx S_2$. This is exactly what we expect from our bisimulation relation since, when the two processes are put in a summation context, their behavior would determine the behavior of the whole context. In fact, the completion of the action in a process $S_1 + CS$ would interrupt any action currently running in CS and, differently, for the case of $S_2 + CS$ no actions in CS would be interrupted. This is due to the fact that the two processes perform the same action but with different prefix operators.

In order to see whether this is important, it is enough to consider the toy example given in the previous section where actions are modeled by a process $C \stackrel{\text{def}}{=} \alpha.(C \mid C) + \gamma : (G, P)$. In this example the creation of two new cells should be an

event which interrupts, when completed, the production of protein P by the cell. Of course, if the process used in the model would have been $C \stackrel{\text{def}}{=} \alpha : (C, C) + \gamma : (G, P)$, then the completion of the duplication process for the cell would not have interrupted the production of P.

4. Conclusions

In this paper we have considered the problem of modeling biological systems in which different scale levels are taken into account resulting in the fact that actions at a higher level may take more time than actions at a lower one.

In order to model such systems we have defined PAPC, a variant of the CCS process algebra in which a process may be simultaneously involved in one long lasting high level action and in several faster lower level actions. In order to model this, we have added in the algebra two different prefix operators to model the role of a process in an action. An action can be considered either as conservative or preemptive in a process, resulting in two different behaviors for the process and for the other actions which are already started and not completed.

We have given both a reduction semantics and a compositional Structural Operations Semantics for PAPC by means of different relations, one for each of the possible events which may change the state of a process, namely the start and the completion of the actions. The semantics we have given are in ST style as this permits to observe the start and the completion of an action as two detached events. This style of the semantics permits also to observe processes in configurations in which multiple actions are started and not completed.

We have also defined a notion of behavioral equivalence for PAPC processes based on the ideas of higher-order bisimulations for process calculi. We have proved that our bisimulation is a congruence with respect to all PAPC operators.

In the paper, we also showed some simple examples of PAPC processes such that their semantics permits to observe the key features of the algebra. We also discussed the notion of bisimulation we introduced by analyzing two simple PAPC processes.

As future work we will apply PAPC to the modeling of multiscale systems in order to prove the utility of the formalism. To this purpose we may consider enriching PAPC with biologically inspired operators to easily model complexation, decomplexation or more complex biological structures as membranes or compartments as it has been previously done with other calculi. Moreover, we may consider the definition of more biologically inspired notions of equivalence for PAPC processes (see e.g., [28,29]). Finally, we may replace our unquantified delays with probabilistic timed delays with general distributions. To this purpose, it is worth noting that in [17] it is showed how ST approach easily supports such a feature.

References

[1] L. Aceto, W.J. Fokkink, C. Verhoef, Structural operational semantics, in: J.A. Bergstra, A. Ponse, S.A. Smolka (Eds.), Handbook of Process Algebra, Elsevier, 2001, pp. 197–292. Chapter.
[2] T. Alarcon, H.M. Byrne, P.K. Maini, A multiple scale model for tumour growth, Multiscale Model. Simul. 3 (2005) 440–475.
[3] E. Astesiano, A. Giovini, G. Reggio, Generalized bisimulation in relational specifications, in: Proc. STACS 98, in: LNCS, vol. 294, Springer, 1988, pp. 207–226.
[4] G.S. Ayton, W.G. Noid, G.A. Voth, Multiscale modeling of biomolecular systems: in serial and in parallel, Curr. Opin. Struct. Biol. 17 (2) (2007) 192–198.
[5] R. Barbuti, G. Caravagna, A. Maggiolo-Schettini, P. Milazzo, On the interpretation of delays in delay stochastic simulation of biological systems, in: 2nd Int. Workshop on Computational Models for Cell Processes (CompMod'09), EPTCS 6, 2009, pp. 17–29.
[6] R. Barbuti, G. Caravagna, A. Maggiolo-Schettini, P. Milazzo, Delay stochastic simulation of biological systems: a purely delayed approach, in: C. Priami, et al. (Eds.), Transactions on Computational Systems Biology XIII, in: LNBI, vol. 6575, 2011, pp. 61–84.
[7] R. Barbuti, G. Caravagna, A. Maggiolo-Schettini, P. Milazzo, G. Pardini, The calculus of looping sequences, in: M. Bernardo, P. Degano, G. Zavattaro (Eds.), Formal Methods for Computational Systems Biology, SFM 2008, in: LNCS, vol. 5016, Springer, 2008, pp. 387–423.
[8] R. Barbuti, G. Caravagna, A. Maggiolo-Schettini, P. Milazzo, S. Tini, Aspects of multiscale modelling in a process algebra for biological systems, in: Int. Meeting on Membrane Computing and Biologically Inspired Process Calculi, MeCBIC'10, EPTCS 40, 2010, pp. 54–69.
[9] R. Barbuti, A. Maggiolo-Schettini, P. Milazzo, G. Pardini, Spatial calculus of looping sequences, in: Int. Workshop From Biology to Concurrency and Back (FBTC'08), ENTCS, 229(1), 2008, pp, 21–39.
[10] R. Barbuti, A. Maggiolo-Schettini, P. Milazzo, S. Tini, Compositional semantics and behavioral equivalences for P Systems, Theor. Comput. Sci. 395 (1) (2008) 77–100.
[11] R. Barbuti, A. Maggiolo-Schettini, P. Milazzo, S. Tini, A P systems flat form preserving step-by-step behaviour, Fundam. Inform. 87 (1) (2008) 1–34.
[12] R. Barbuti, A. Maggiolo-Schettini, P. Milazzo, S. Tini, Compositional semantics of spiking neural P systems, J. Log. Algebr. Program. 79 (6) (2011) 304–316.
[13] R. Barbuti, A. Maggiolo-Schettini, P. Milazzo, S. Tini, An overview on operational semantics in membrane computing, Int. J. Found. Comput. Sci. 22 (1) (2011) 119–131.
[14] F. Billy, B. Ribba, O. Saut, H. Morre-Trouilhet, T. Colin, D. Bresch, J.P. Boissel, E. Grenier, J.P. Flandrois, A pharmacologically based multiscale mathematical model of angiogenesis and its use in investigating the efficacy of a new cancer treatment strategy, J. Theoret. Biol. 260 (2009) 545–562.
[15] G. Boudol, Towards a lambda-calculus for concurrent and communicating systems, in: Proc. TAPSOFT, in: LNCS, vol. 351, Springer, 1989, pp. 149–161.
[16] M. Bravetti, R. Gorrieri, Deciding and axiomatizing st bisimulations for a process algebra with recursion and action refinement, Tech, Rep. UBLCS-99-1, University of Bologna, 1999.
[17] M. Bravetti, R. Gorrieri, The theory of interactive generalized semi-Markov processes, Theor. Comput. Sci. 282 (1) (2002) 5–32.
[18] L. Brodo, P. Degano, C. Priami, A stochastic semantics for BioAmbients, in: Proceedings of PaCT07, in: LNCS, Springer, 2007, pp. 22–34.
[19] H.M. Byrne, M.R. Owen, T. Alarcon, J. Murphy, P.K. Maini, Modelling the response of vascular tumours to chemotherapy: a multiscale approach, Math. Mod. Meth. Appli. Sci. 15 (2006) 1219–1241.
[20] G. Caravagna, Formal modeling and simulation of biological systems with delays, Ph.D. Thesis, Department of Computer Science, University of Pisa, 2011.
[21] L. Cardelli, On process rate semantics, Theoret. Comput. Sci. 391 (3) (2008) 190–215.
[22] F. Ciocchetta, J. Hillston, Bio-PEPA: a framework for the modelling and analysis of biochemical networks, Theoret. Comput. Sci. 410 (33-34) (2009) 3065–3084.

[23] F. Ciocchetta, J. Hillston, Calculi for biological systems, in: M. Bernardo, P. Degano, G. Zavattaro (Eds.), Formal Methods for Computational Systems Biology (SFM 2008), in: LNCS, vol. 5016, Springer, 2008, pp. 265–312.
[24] R. Cleaveland, G. Lüttgen, V. Natarajan, Priority in process algebra, in: J.A. Bergstra, A. Ponse, S.A. Smolka (Eds.), Handbook of Process Algebra, Elsevier, 2001, pp. 711–765.
[25] V. Danos, J. Feret, W. Fontana, R. Harmer, J. Krivine, Rule-based modelling of cellular signalling, in: Proceedings of CONCUR07, Springer, LNCS, 2007, pp. 17–41.
[26] A. Degasperi, M. Calder, Process algebra with hooks for models of pattern formation, in: PROC. CS2Bio 2010, 2010, pp. 31–47.
[27] A. Degasperi, M. Calder, Multiscale modelling of biological systems in process algebra with multi-way syncrhonization, in: PROC. CMSB 11, 2011.
[28] V. Galpin, J. Hillston, F. Ciocchetta, A semantic equivalence for Bio-PEPA based on discretisation of continuous values, Theoret. Comput. Sci. 421 (2011) 2142–2161.
[29] V. Galpin, J. Hillston, F. Ciocchetta, A semi-quantitative equivalence for abstracting from fast reactions, in: PROC. CompMod, 2011, pp. 34–49.
[30] D. Gillespie, Exact stochastic simulation of coupled chemical reactions, Journal of Physical Chemistry 81 (1977) 2340.
[31] M. Hennessy, Axiomatising finite concurrent processes, SIAM Journal of Computing 17 (5) (1988) 997–1017.
[32] V. Kumar, A.K. Abbas, N. Fausto, J. Aster, Tissue Renewal, Regeneration, and Repair, 8th Edition, in: Robbins and Cotran Pathologic Basis of Disease, Elsevier Saunders, 2010.
[33] M. Kwiatkowski, I. Stark, The continuous Π-calculus: a process algebra for biochemical modelling, in: Proceedings of CMSB08, in: LNCS, vol. 5307, Springer, 2008, pp. 103–122.
[34] R.J. van Glabbeek, The refinement theorem for st bisimulation semantics, in: Proc. IFIP Working Conference on Programming Concepts and Methods, North Holland, 1990.
[35] R.J. van Glabbeek, F.W. Vaandrager, Petri net models for algebraic theories of concurrency, in: Proc. PARLE, in: LNCS, vol. 259, Springer, 1987, pp. 224–242.
[36] H. Lodish, et al., Molecular Cell Biology, 4th Edition, W.H. Freeman, New York, 2000.
[37] P. Milazzo, Qualitative and quantitative formal modeling of biological systems, Ph.D. Thesis, Department of Computer Science, University of Pisa, 2007.
[38] R. Milner, Communication and Concurrency, in: International Series in Computer Science, Prentice Hall, ISBN: 0-131-15007-3, 1989.
[39] R. Milner, J. Parrow, D. Walker, A calculus of mobile processes, I, Inf. Comput. 100 (1) (1992) 1–40.
[40] R. Milner, J. Parrow, D. Walker, A calculus of mobile processes, II, Inf. Comput. 100 (1) (1992) 41–77.
[41] M. Mousavi, M. Gabbay, M.A. Reniers, SOS for higher order processes, in: Proc. CONCUR 2005, in: LNCS, vol. 3653, Springer, 2005, pp. 308–322.
[42] M. Pedersen, G. Plotkin, A language for biochemical systems: design and formal specification, Transactions on Computational Systems Biology XII 5945 (3) (2010) 77–145.
[43] G.D. Plotkin, A structural approach to operational semantics. Tech. Rep. DAIMI FN-19, Aarhus University, Denmark, 1981.
[44] C. Priami, Stochastic π-calculus, The Computer Journal 38 (7) (1995) 578–589.
[45] C. Priami, A. Regev, E. Shapiro, W. Silverman, Application of a stochastic name-passing calculus to representation and simulation of molecular processes, Information Processing Letters 80 (2001) 25–31.
[46] A. Regev, E.M. Panina, W. Silverman, L. Cardelli, E.Y. Shapiro, Bioambients: an abstraction for biological compartments, Theoretical Computer Science 325 (1) (2004) 141–167.
[47] A. Regev, W. Silverman, E. Shapiro, Representation and simulation of biochemical processes using the pi-calculus process algebra, in: Pacific Symposium on Biocomputing, vol. 6, World Scientific Press, 2001, pp. 459–470.
[48] B. Ribba, T. Colin, S. Schnell, A multiscale mathematical model of cancer, and its use in analyzing irradiation therapies, Theor. Biol. Med. Model 3 (7) (2006).
[49] P.M.A. Sloot, A.G. Hoekstra, Multi-scale modelling in computational biomedicine, Brief. Bioinform. (2009) doi:10.1093/bib/bbp038.
[50] B. Thomsen, A theory of higher order communicating systems, Inform. Comput. 116 (1995) 38–57.